Time Synchronization in the DHT-based P2P Network Kad for Real-Time Automation Scenarios

Jan Skodzik, Peter Danielis, Vlado Altmann, Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany, Tel./Fax: +49 381 498-7284 / -1187251
Email: jan.skodzik@uni-rostock.de

***Abstract*—In this paper, an approach to synchronize the P2P network Kad to be applied in automation scenarios is presented. The approach bases on a deterministic algorithm to synchronize the network, which is required for hard real-time applications. Today's Industrial Ethernet solutions base on centralized structures, which are deficient in resilience and scalability regarding network administration and size. The presented decentralized approach benefits from nodes helping to synchronize the network. However, the higher the number of helping nodes the higher is the time deviation on the nodes of the network, which contrary results in a higher time error. Therefore, a trade-off between synchronization performance and time error has to be determined to meet predefined constraints depending on the application scenario. Moreover, the individual clock drift of every device is considered to define necessary re-synchronization intervals of the network. Additionally, the optimum number of nodes to synchronize the Kad based network has been identified and the resulting synchronization performance and generated traffic are determined. Furthermore, an approach is presented to handle the dynamic churn of nodes.**

***Keywords*-Peer-to-Peer; Kad; Synchronization; Automation;**

I. Introduction

In the field of automation, there is a high need for hard real-time systems to guarantee the proper function of, e.g., production sites. There are several realizations, which use common Ethernet technology with proprietary hardware and protocols to ensure hard real-time communication. Industrial Ethernet solutions like Profinet, EtherCat, Modbus-TCP, or Powerlink have central instances to manage the communication between the network nodes [1]–[4]. They use a master/slave or client/server model with central components representing a single point of failure (SPOF). The failing of the central instance leads to the total failing of system functionality and the loss of data. Furthermore, the physical integration of new nodes into the infrastructure and integration into the managing software require high efforts. The central instance has to monitor the whole communication and thus is a bottleneck in the network. Another disadvantage is the network topology of Industrial Ethernet solution. The network topology is often limited to ring or line structures. Sometimes it is mentioned that the physical topology does not require a ring or line structure but the virtual topology does. Additionally, in traditional facilities the structure is organized hierarchically for better maintenance, which prevents direct communication between devices at the lower hierarchical level. Contrary, P2P networks solve the problem of limited scalability, low robustness, and lacking user friendly installation. Each peer provides server and client functionality, which results in a network with high resilience. Kad (an implementation of Kademlia) has been chosen as network to realize a self-organizing distributed hash table (DHT)-based P2P network. A network consisting of several subnets should act as one P2P network realizing a total horizontal and vertical integration of a facility automation network. A main requirement for networks in the field of automation is a common time base for controlled information exchange. This paper presents an approach to realize the synchronization of the Kad network by applying small changes to the Kad protocol. By allowing a defined grade of parallelism in the communication, the time needed for synchronization is decreased while the time error increases. Therefore, a trade-off is investigated in this regard. Additionally, the drift of each device forces the system to re-synchronize the network. Therefore, the consumed time for synchronization has to be kept minimal while considering the time error.

The remainder of this paper is organized as follows: Section II contains a comparison of the proposed approach with related work. Section III presents the necessary steps to realize a Kad-based network synchronization. An algorithm called KaDisSy is presented, which carries out the synchronization. Section IV mathematically analyzes the performance of the presented solution. In Section V, two aspects are discussed to decrease the time error by peer grouping and to increase robustness by a backup approach. The paper concludes in Section VI.

II. Related Work

The Network Time Protocol (NTP) and the Precision Time Protocol (PTP) are two established protocols to synchronize devices in a network. They achieve an accuracy of several milliseconds in case of NTP [5] and about 100 microseconds for a single link in case of PTP [6]. Furthermore, PTP can achieve better performance than NTP as it uses special hardware to generate accurate time stamps, which increases the time resolution and decreases the time error.

Contrary, this paper presents a software-based solution, without any special hardware requirements. Furthermore, NTP and PTP base on a hierarchical approach, which contrasts with the idea of P2P-based networks such as Kad [7], [8].

An alternative synchronization approach called PariSync for P2P networks is presented in [9]. It bases on Java and achieves a synchronization error of a few hundred milliseconds over the Internet. PariSync is suggested to be used in large networks with high churn. The information exchange and media access is not controlled. The resulting error through parallel cross talk and switch buffering is not investigated. Also, it takes many seconds to establish a stable network.

In [10], a P2P system is presented to improve the routing paths in terms of traffic cost reduction inside the network. The main focus is on a large network with high churn. The Network Time Protocol (NTP) is used to synchronize the nodes. However, a time resolution and average time error are not mentioned. Additionally, the traffic itself is not managed, which leads to further time errors.

In [11], a system is presented, which uses a gossip-based approach. The focus is on assessing the impact of corrupted processes on the effectiveness of clock synchronization. There is no formal proof of the correctness of clock synchronization convergence. [12] presents a system using aggregation to calculate average, product, and extreme values in P2P networks with an epidemic approach. This requires real-time intervals called cycles to generate global values by the systems. However, these cycles depend on an appropriate synchronization of the nodes as well. The negative effect of parallel communication due to time errors is not considered and the drift and message delays are only discussed informally. Several parameters have to be determined to achieve a high probability of information exchange. Using a gossip-based or epidemic approach like in [11] and [12] is always a trade-off between scalability and reliability [13]. Therefore, we renounce a gossip-based approach as it is probabilistic and not suited for hard real-time scenarios requiring deterministic behavior of the network.

The goal is to achieve a similar time resolution and time error like NTP and software-based PTP protocol. We present an effective trade-off between exclusive media access and time error increase by speeding up the synchronization process enabled by parallel communication of nodes. The presented approach does not require any central instance and offers a high performance synchronization of several thousand devices with a low time error.

III. Basics and design concept

A deterministic approach to synchronize the Kad network is presented consisting of six stages. These stages represent a working system based on a Kad network for an automation scenario and are depicted in Figure 1. Following, the steps are elaborated to determine the worst case for the performance of the presented approach. Kad has been chosen because it does not have any SPOF as it belongs to the fully decentralized structured P2P systems and offers the best lookup performance among the DHT-based solutions with $O(log_{2^b}(N))$ [14]. The network is designated to run in an automation environment. Additionally, the realization of the presented approach will run on a dedicated network. Thus, no generated competing data traffic by other components is assumed.

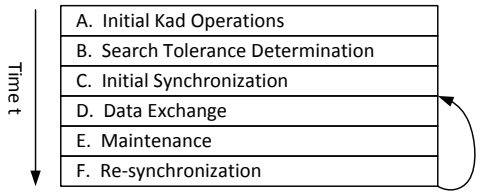

Fig. 1. Synchronization stages

A. Initial Kad Operations

During the first stage, the Kad network performs initial operations. This includes the bootstrapping of new nodes and the maintenance of the network thereby filling the routing tables with contacts. Additionally, to avoid further message overhead in the following stages the MAC addresses of the nodes are also stored to avoid Address Resolution Protocol (ARP) packets in IPv4 or Neighbor Discovery Protocol (NDP) packets in IPv6 networks. This is necessary as an operating system's ARP cache is limited in space and the entries will be stored for a certain period of time. If there is no available entry an ARP broadcast in the network is necessary, which could result in heavy performance losses and increased time error.

B. Search Tolerance Determination

To ensure that all lookup processes are successful, the dynamic search tolerance (DST) was developed [15]. The DST algorithm adjusts the search tolerance at runtime so that at least one node is responsible for every hash value. Moreover, this increases the performance as it leads to less time outs terminating the lookup process during the Kad operations. The DST algorithm is started by any node in the network receiving a defined trigger. The node receiving the trigger command is called the first triggered (FT) node.

C. Initial Synchronization

After the FT node has performed the DST process, the initial synchronization process is executed. The FT node sends a broadcast through the network and has to wait for at least the time a packet travels the critical path in the network. The waiting time must be chosen in such a way that every node receives the broadcast message. Now, the FT node exclusively accesses the Ethernet medium.

Kad applies the iterative lookup strategy and originally the active node would be able to issue α lookup requests in parallel [16]. However, to ensure exclusive Ethernet access, only one lookup request in the network at one time is allowed. Thus, α is set to 1. During the synchronization process, the maintenance is paused and the routing table is only modified by the lookup process. This does not pose a problem as in the original implementation, maintenance operations are executed every several ten seconds. The synchronization is done by the Kademlia Discovery and Synchronization (KaDisSy) algorithm, which is a version of the Kademlia Discovery algorithm extended by a synchronization functionality [17]. It

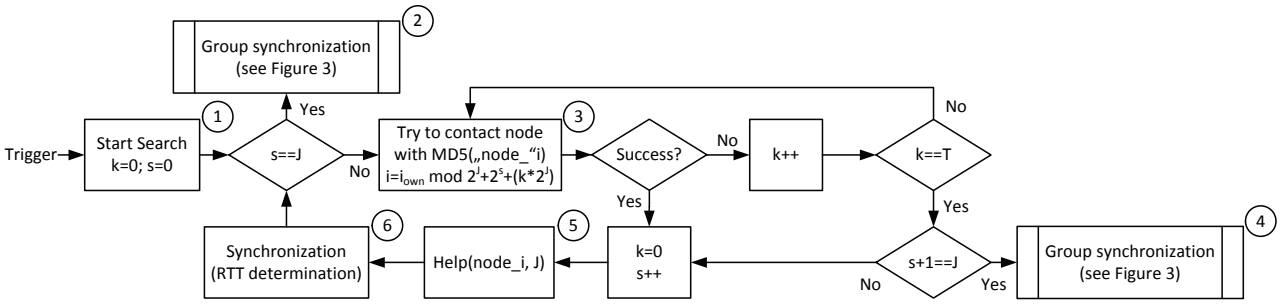

Fig. 2. KaDisSy algorithm: Acquiring HT nodes

is assumed that the devices in the Kad network follow naming conventions. That is, every node's hash value is calculated from a concatenation of a fixed string and a counter value i. This offers the possibility to operate independently of the hash value distribution. The algorithm consists of two steps as follows:

Acquiring helping nodes: The first step is the acquiring of helping triggered (HT) nodes and is depicted in Figure 2. With every step the number of HT nodes doubles as every HT node itself requests new HT nodes. At the start (1), the counters are set to their initial values. s counts the number of steps to acquire HT nodes. It is possible to set J to zero. Then, the FT node is the only node responsible for the synchronization. In this case, the algorithm immediately continues with the group synchronization (2), which is described later. However, J is usually set to a higher value to acquire HT nodes. Therefore, the total number of nodes responsible for the synchronization is 2^J. If s is smaller than J the FT or HT node tries to contact the next node to request it as HT node (3). The hash value as a base of the lookup process is calculated from a user pre-defined concatenation of a string, e.g., "node_" and an integer value i. To create the hash value, the Message-Digest 5 algorithm is used. i is calculated from the value i_{own}, which is the i of the requesting node, J, s, and the counter k (see Formula 1).

$$i = i_{own} \bmod 2^J + 2^s + (k * 2^J) \tag{1}$$

If the responsible node for the value i is not found by the lookup process as it is not existent in the Kad network the offset counter k is increased. If k is smaller than the threshold value T a new lookup (3) is performed. Otherwise, the node checks if $s+1$ equals J, which indicates that this is the last iteration for acquiring new HT nodes. If it is the last iteration the algorithm starts with the group synchronization (4). Otherwise, k is reset to zero and s is incremented by 1 to request a new HT node for the next group.

Contrary, if the previous lookup with the value i was successful k is reset and s is incremented. The node is requested to be a HT node and gets to know the parameter J (5). Now, the synchronization process (6) is started by sending a ping from the requesting node to the new HT node to determine the round trip time (RTT). Due to channel symmetry in an automation Kad network, the delay between two nodes is assumed to be equal in both directions. The determined delay $RTT/2$ will be added to the actual time of the FT or HT node and sent via UDP packets to the node, which has to be synchronized. The new HT node needs to determine its initial counter value s for the next iteration. This value does not need to be given by the requesting node as is can be determined by Formula 2.

$$s = \lfloor log_2(i_{own} \bmod 2^J) + 1 \rfloor \tag{2}$$

The algorithm starts again to search for further HT nodes as long as s is smaller than J or until k is equal T and it is the last iteration of acquiring new HT nodes. After that, the algorithm proceeds with group synchronization (4).

Group Synchronization: The second synchronization step consists in the synchronization of the own group (see Figure 3). The FT node and each HT node synchronize their nodes, which are potentially in its group. Similar to the acquiring

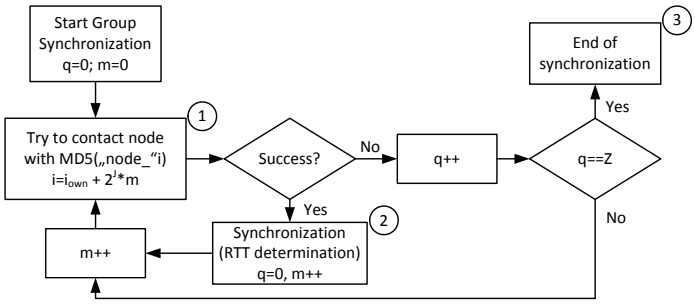

Fig. 3. KaDisSy algorithm: Group synchronization

process, the FT and each HT node contact the nodes in their group depending on their i_{own} value plus an offset depending on m (1). Additionally, if the lookup is not successful the counters q and m are increased as long as q is smaller than the threshold value Z. In this case, the synchronization process is finished (3) as it is assumed that there are no more nodes in the group. If the lookup is successful q is reset to zero and the synchronization between the FT or HT node and the requested node is performed (2), which is similar to the synchronization during the HT node acquiring step.

D. Data Exchange

During data exchange between the nodes, no new nodes are allowed to enter the Kad automation network as this would

result in a higher time error. Additionally, the data exchange amounts to the largest share of the time periods as it represents the main functionality of the Kad network. The communication including the data exchange between the nodes has to be handled by a separate protocol, which is not part of this work.

E. Maintenance

The maintenance and the entering of new nodes into the Kad network were forbidden during the previous stages to avoid increasing time errors. However, to keep the routing tables updated and to allow the entering of new nodes the maintenance stage is required. Nodes are allowed to join the network by requesting their bootstrap node after receiving a jamming signal, which is a broadcast issued by the FT node through the Kad network. The mentioned requirements can be realized by running the original Kad network implementation.

F. Re-synchronization

Due to their clock drift, the nodes need to re-synchronize periodically. The re-synchronization period T_{ReSyn} is defined by Formula 3 and represents the time between two synchronizations. The maximum allowed time error is given with $T_{MaxError}$. $T_{SynError}$ is the error during the synchronization process between two nodes. The value depends on the grade of parallel communication as there can be buffered packets in the network, which adulterate the synchronization time. D_{Clk} is the drift of the clocks of the devices. T_{ReSyn} directly depends on $T_{SynComp}$ as the first node already drifted away from the correct clock value while the last node is synchronized.

$$T_{ReSyn} < \frac{T_{MaxError} - T_{SynError}}{2 * D_{Clk}} - T_{SynComp} \quad (3)$$

A detailed analysis of the performance can be found in Section IV. The re-synchronization is technically similar to the initial synchronization. After finishing the re-synchronization, we proceed with the data exchange again and repeat the following stages.

IV. Performance Analysis

As the KaDisSy algorithm is supposed to be used in a hard real-time environment, it is necessary to determine its worst case behavior.

Synchronization performance: First, the time T_{Syn} needed to synchronize two nodes is defined by Formula 4. N is the number of nodes in the Kad network. b describes how many bits can be skipped by each lookup step and is set to 1 for worst case analysis. $T_{SynComp}$ in Formula 5 is the time to complete the synchronization process for all nodes and directly depends on N and the number of iterations J for acquiring HT nodes.

$$T_{Syn} = \lceil log_{2^b}(N) \rceil * RTT + 1.5RTT \quad (4)$$

$$T_{SynComp} = T_{Syn} * (J + \frac{N}{2^J} - 1) \quad (5)$$

The time consumption $T_{SynComp}$ depending on the number of nodes N and J as an indicator for the number of HT nodes is depicted in Figure 4.

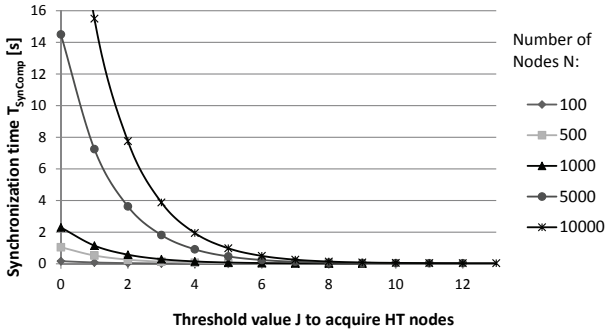

Fig. 4. Time to sychnronize the Kad network

Consideration of queuing delay: If there are no HT nodes helping during the synchronization process the time for synchronization linearly depends on the number of nodes in the network and does not scale well. Otherwise, if $T_{SynComp}$ must be kept low due to application-specific requirements, it is possible to increase the number of helping HT nodes. If the number of helping nodes increases, the general timing error increases as well due to parallel packet exchange in the network. In a switched Ethernet network environment, the queuing delay of packets has to be considered as it increases the synchronization error. The Kad packets are the largest packets and therefore decisive for the worst case analysis. T_{Pkt} therefore represents the time to transmit a packet through a switch. The worst case for the synchronization time error $T_{SynError}$ in a switched Ethernet network is described by Formula 6. $T_{SynError}$ linearly depends on the number of HT nodes. The synchronization deviation $T_{SynDeviation}$ is given with 30 μs, which is the measured worst case synchronization deviation between two dedicated PCs and has been determined by practical investigations.

$$T_{SynError} = T_{SynDeviation} + (\#HT * T_{Pkt}) \quad (6)$$

The time T_{ReSyn} is defined by Formula 3 and should be as large as possible to provide sufficient time for data exchange. Several parameters have to be declared to determine proper values. The maximum allowed time error is given by $T_{MaxError}$. It is set to 1 ms for all further considerations. It is assumed that the network consists of a cascaded switched network. A Kad response packet with a size of 80 byte, which is the biggest packet possible, containing a fixed number of contacts took 610 ns to traverse the switch as a consistent network using 1 Gbit/s connections is assumed. Every queued packet has to wait until the previously queued packet is processed. Therefore, T_{Pkt} is set to 610 ns for the worst case analysis.

Finally, it is necessary to know the time period for the synchronization. All important times are listed in Table I for reasons of clarity. Additionally, the clock is not an ideal electronic part in real world conditions. Each clock has a

SYMBOL	DESCRIPTION
T_{Syn}	Time to synchronize two nodes
$T_{SynComp}$	Time to synchronize all nodes
$T_{SynError}$	Synchronization error between two nodes
T_{Pkt}	Time to traverse 1 Gbit/s Switch
$T_{MaxError}$	Maximum allowed time error in the network
T_{ReSyn}	Time interval between two synchronization steps

TABLE I
IMPORTANT TIMES FOR PERFORMANCE EVALUATION

unique drift. As the target platform for a future prototype, the Zedboard is intended. The internal oscillator provides a stable clock with a drift D_{Clk} of $\pm$ 50 ppm [18]. In Figure 5, the time period T_{ReSyn} depending on the number of nodes N and HT nodes is apparent. It is useful to add HT nodes to gain a better performance and also to support a higher number of nodes in the Kad automation network. With 10,000 nodes, it is apparent that more HT nodes are needed to synchronize the network in an adequate time.

If T_{ReSyn} is zero the synchronization has to be executed permanently. Additionally, it is apparent if too many HT nodes are created, due to a large J the possible worst case error becomes too dominant and the re-synchronization has to be executed more often. As apparent, in larger Kad automation networks, one FT node is not able to synchronize the network as T_{ReSyn} must be greater than $T_{SynComp}$. Therefore, HT nodes are necessary. In the best case, their number must be set to such a value that T_{ReSyn} becomes much greater than $T_{SynComp}$ so that the synchronization process is not the dominating process. This optimum value for J is J_{Opt} defined by Formula 7 and

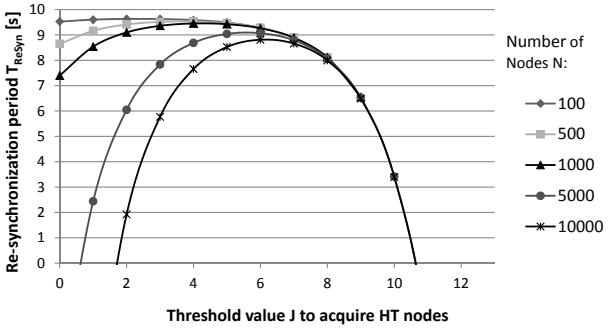

Fig. 5. Minimal Synchronization Period

represents the maximum value for T_{ReSyn}, which results in less need for re-synchronization and more time for data exchange. J_{Opt} depends on the number of nodes N in the network, T_{Syn} denoting the time needed to synchronize two nodes, and T_{Pkt}.

Furthermore, it is important to know the traffic generated by the presented approach. To determine the traffic volumes, the packet types must be known, which are exchanged. First, this includes the Kad response packet with a size of 80 byte as mentioned above. Additionally, Kademlia requests (35 byte in size), two packets for pinging (each 20 byte in size), and one packet containing the 4 byte time value (24 byte in size, see step (6) in Figure 2) are involved in the synchronization process.

$$J_{Opt} = \log_2 \left(\frac{D_{Clk} * T_{Syn} \left(\sqrt{1 + \frac{(2ln(2))^2 * N * T_{Pkt}}{2 D_{Clk} * T_{Syn}}} - 1 \right)}{ln(2) * T_{Pkt}} \right) \quad (7)$$

In Table II, characteristic values are given for rounded J_{Opt} values for different network sizes. As apparent, the performance

NODES	100	500	1000	5000	10000
J_{Opt}	3	4	5	6	7
$T_{SynComp}$ [ms]	24.65	71.93	81.08	241.06	260.79
T_{ReSyn} [s]	9.63	9.53	9.42	9.07	8.66
Traffic [Kbyte]	84	535	1,184	7,610	16,346

TABLE II
PERFORMANCE EVALUATION USING J_{Opt}

is competitive to existing central solutions like NTP with an accuracy of several milliseconds whereby a maximum time error $T_{MaxError}$ of 1 ms is assumed. Furthermore, our approach is not gossip-based like the solutions in [11] and [12] and therefore it is suitable to be applied in a hard real-time scenario. The results have been proven by simulations and confirm the values for $T_{SynComp}$ and T_{ReSyn}, which have been previously calculated.

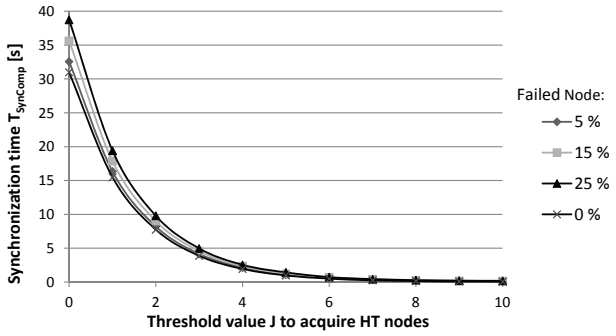

Fig. 6. Synchronization performance considering failed nodes

Performance impact due to failed nodes: Additionally, it is necessary to know how the algorithm acts if we consider the churn of nodes. Therefore, the performance has been traced with different failure rates of the nodes ranging from 1 % to 25 % percent whereby 25 % is a very pessimistic assumption. The size of the network is set to 10,000 nodes. If a HT node is not able to synchronize with a failed node a timeout is generated after 6200 μs, which is twice as long as the time to synchronize two nodes in a network with 10,000 nodes. The parameters T and Z from the KaDisSy algorithm are set to the value 10. All available nodes were synchronized properly. Exemplary simulation results for failure rates of 5%, 15%, and 25% compared to a network without failed nodes are depicted in Figure 6. Additionally, the deviation of the synchronization performance $T_{SynComp}$ compared to a network without failed nodes is given in Figure 7. As apparent, the

failure rate has a significant influence on the performance and should be considered if the network characteristics are known in advanced. If a failure rate of 25 % and 10,000 nodes are assumed $T_{SynComp}$ increases by 25 % (using a single node to synchronize, i.e., $J = 0$) to 100 % (using $J = J_{Opt}$) compared to the network without failed nodes.

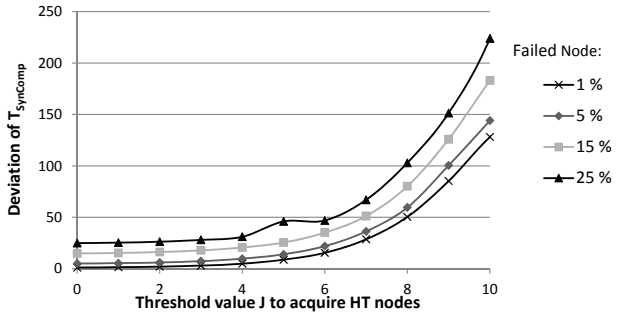

Fig. 7. Deviation of $T_{SynComp}$ compared to a network without failed nodes

V. Further improvements

Following, two approaches are presented to improve the presented solution by decreasing the time error and increasing reliability.

The grouping of nets: If the P2P network consists of several subnets it is possible to determine a corresponding HT node in each subnet to realize the synchronization inside it. Furthermore, this could increase the accuracy of the synchronization due to less time errors as the packets to forward the times values are exchanged within the subnet. To address the nodes in each single subnet, a consistent naming convention for each subnet has to be given. However, subnets should work independently. Otherwise, small subnets have to wait long for the completion of the synchronization of large subnets.

The backup system: Another focus is on ensuring high reliability for a proper working network especially by using backup concepts in the automation area. In traditional systems, a wide variety of parameters, algorithms, or user defined constraints have to be exchanged to determine a backup node, which leads to additional communication overhead. Contrary, in the presented approach the problem of a failing initial FT or HT node is solved without heavy overhead. If the FT node, which can be any node in the Kad network, fails the closest node in terms of the XOR distance to zero in the hash table becomes responsible for the initial synchronization stage. A node can calculate the worst case time, in which it should be contacted. If this time is exceeded the node becomes active and starts the (re-)synchronization.

VI. Conclusion and Future Work

An approach to synchronize the DHT-based P2P network Kad is presented to enable its operation in real-time automation scenarios. A worst case analysis has been carried out to show the performance, which is close to existing software solutions like NTP or PTP, but avoids SPOFs in the form of central instances. Additionally, the optimum number of helping nodes is defined as a trade-off between fast synchronization and less need for re-synchronization due to time errors and drifts of the clocks. The idea of grouping inside the network could lead to better results due to bounded subnets and less time error variations. Also, the handling of failed nodes and new entering nodes into the network (called churn) has been considered and an approach to handle the node churn is presented. Prospectively, a real-time Kad prototype will be developed to demonstrate the performance of the synchronization algorithm on existing hardware and enable further automation applications.

References

[1] PROFIBUS Nutzerorganisation e.V. (2012) Profinet. [Online]. Available: http://www.profibus.com/technology/profinet/

[2] EtherCat Technology Group. (2012) Ethernet for control automation technology. [Online]. Available: http://www.ethercat.org/

[3] M. Organization, "Modbus specifications," 2013. [Online]. Available: http://www.modbus.org/

[4] EPSG. (2012) Ethernet powerlink. [Online]. Available: http://www.ethernet-powerlink.org

[5] U. Windl, D. Dalton, Hewlett-Packard, M. Martinec, J. S. Institute, and D. R. Worley. (2006) The ntp faq and howto. [Online]. Available: http://www.ntp.org/ntpfaq/NTP-s-algo.htm

[6] D. Seely, "AN-1728 IEEE 1588 Precision Time Protocol Time Synchronization Performance," Texas Instruments, Tech. Rep., 2007. [Online]. Available: http://www.ti.com/lit/an/snla098/snla098.pdf

[7] D. Mills, "Network Time Protocol (NTP)," RFC 1305, Internet Engineering Task Force, March 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1305.txt

[8] K. Lee and J. Eidson, "IEEE-1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," in *In 34 th Annual Precise Time and Time Interval (PTTI) Meeting*, 2002, pp. 98–105.

[9] P. Bertasi, M. Bonazza, N. Moretti, and E. Peserico, "PariSync: Clock synchronization in P2P networks," in *Precision Clock Synchronization for Measurement, Control and Communication, 2009. ISPCS 2009. International Symposium on*, oct. 2009, pp. 1 –6.

[10] Y. Liu, L. Xiao, and L. Ni, "Building a scalable bipartite P2P overlay network," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, april 2004, p. 46.

[11] R. Baldoni, M. Platania, L. Querzoni, and S. Scipioni, "A Peer-to-Peer Filter-Based Algorithm for Internal Clock Synchronization in Presence of Corrupted Processes," in *Dependable Computing, 2008. PRDC '08. 14th IEEE Pacific Rim International Symposium on*, dec. 2008, pp. 64 –72.

[12] A. Montresor, M. Jelasity, and O. Babaoglu, "Robust aggregation protocols for large-scale overlay networks," in *Dependable Systems and Networks, 2004 International Conference on*, june-1 july 2004, pp. 19 – 28.

[13] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "Epidemic information dissemination in distributed systems," *Computer*, vol. 37, no. 5, pp. 60 – 67, may 2004.

[14] K. Steinmetz, Ralf; Wehrle, *Peer-to-Peer Systems and Applications*. Springer, 2005.

[15] P. Danielis, J. Skodzik, V. Altmann, S. Kruse, and D. Timmermann, "Dynamic Search Tolerance for Deterministic Lookup in the DHT-based P2P network Kad at Runtime," in *19th International European Conference on Parallel and Distributed Computing (Euro-Par 2013)*, 2013 (subm.).

[16] D. Stutzbach and R. Rejaie, "Improving Lookup Performance Over a Widely-Deployed DHT," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, april 2006, pp. 1 –12.

[17] J. Skodzik, P. Danielis, V. Altmann, J. Rohrbeck, D. Timmermann, T. Bahls, and D. Duchow, "DuDE: A distributed computing system using a decentralized P2P environment," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, oct. 2011, pp. 1048 –1055.

[18] "Zedboard documentations," 2013. [Online]. Available: http://www.zedboard.org/documentation