HaRTKad: A P2P-based Concept for Deterministic Communication and Its Limitations

Björn Konieczek, Jan Skodzik, Peter Danielis, Vlado Altmann, Michael Rethfeldt, Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany, Tel./Fax: +49 381 498-7269
Email: bjoern.konieczek@uni-rostock.de

***Abstract*—Real-time systems play a major role in the realm of industrial automation. It is predicted for the number of smart interconnected devices that participate in such systems to grow significantly in the future. This development is also referred to as Industrial Internet of Things (IIoT) or Industry 4.0. The high number of devices results in highly distributed applications. Therefore, it is no longer sufficient for each device to be real-time capable. In fact, the influence of the communication on the overall timing behavior of the applications grows. Taking this into account, a variety of real-time capable Ethernet approaches called Industrial Ethernet (IE) emerged. However, the established IE solutions rely on proprietary hardware and/or non standard conform protocol adaptations. This leads to very expensive hardware and incompatibilities with other IE solutions or common Ethernet, and thus degrades the interoperability. HaRTKad describes a purely software-based P2P approach that allows deterministic communication over common Ethernet. Although it solely relies on well-known standards and allows real-time communication, it suffers from a multitude of problems. In this paper, the low network utilization, the handling of hash collisions and the traffic prioritization are revealed as the most significant limitations of HaRTKad and possible solutions to these problems are presented.**

I. Introduction

In the area of industrial automation, fieldbus systems pose a well-established solution for real-time device communication. They not only enable a deterministic message exchange but also satisfy the hard real-time requirements described in [1]. However, fieldbuses lack scalability as they are drastically limited in address space. Furthermore, the different solutions are not compatible with each other leading to a low degree of interoperability. To overcome the weaknesses of such fieldbus systems and allow a total horizontal and vertical integration of automation systems, several Ethernet-based solutions called Industrial Ethernet (IE) emerged [2]. They can satisfy hard real-time requirements like fieldbuses, while allowing a much higher number of network participants. The number of nodes in the network is theoretically only limited by the available MAC or IPv4/v6 address space. Furthermore, IE solutions are much more flexible in terms of network topology, whereas fieldbus systems are typically limited to line or ring structures. Additionally, IE systems have a major economical advantage compared to fieldbuses. The use of Ethernet technologies leads to highly reduced costs for development, production and deployment of such systems and allows for a total vertical and horizontal integration [3]. In this way, industrial facilities can be directly integrated into corporate processes. However, despite of the many advantages, IE systems also suffer from some weaknesses. The majority of the available solutions are based on a master-slave or client-server approach. This implies the necessity of a central instance that poses a single point of failure (SPoF) and a bottleneck. Furthermore, some realizations require dedicated and expensive hardware or rely on non standard conform protocol adaptations to provide the hard real-time behavior. This degrades the flexibility and interoperability between the different solutions. These weaknesses will take on greater significance in the future as the number of intelligent and interconnected devices in industrial facilities will increase drastically [4]. To overcome the described problems, Skodzik et al. proposed Peer-to-Peer (P2P) networks as an alternative to the master-slave or client-server based approaches applied by other IE solutions. This approach omits modifications on lower layers, as P2P networks are realized solely on the application layer. Thus, no special hardware or protocol adaptations are needed. In order to prove the feasibility of this approach, Skodzik et al. developed a hard real-time capable P2P system based on the Kademlia protocol, called HaRTKad [5]. The main focus of HaRTKad is to increase the network's robustness and scalability while only utilizing common Ethernet technologies and maintaining hard real-time capabilities. Although the experimental results in [5], [6] and [7] show the general feasibility of the concept, several new problems arise. In this paper, we reveal the most significant limitations of HaRTKad and propose possible solutions to the described problems. The main contributions are:

- Identification of the limitations of HaRTKad
- Proposal of an algorithm to handle hash collisions in DHT-based P2P networks
- Description of a method to enable traffic prioritization in HaRTKad and increase the maximum number of network participants

The remainder of this paper is organized as follows: In Section II, the related work in the area of real-time communication systems is presented. Section III describes the basic mechanisms of HaRTKad and Section IV outlines the most significant weak spots of HaRTKad. In Section V, we present a way to cope with hash collisions. Section VI, describes an

approach to enable traffic prioritization in HaRTKad. Finally, the paper concludes in Section VII.

II. Related Work

Traditional fieldbus systems like PROFIBUS or CC-Link are gradually replaced by Industrial Ethernet to allow a higher number of interconnected devices and increase interoperability with the remaining company infrastructure. [1] gives a summary of current IE solutions and categorizes them according to their real-time capabilities and hardware/software requirements. The majority of current IE realizations like PROFINET IO/IRT, EtherCAT or SERCOS III rely on the master-slave or client-server principle, which implies the necessity of a central instance [8] [9] [10]. EtherCAT, for example, needs a central master to synchronize the connected devices and control the information exchange. This central instance poses a SPoF and bottleneck. Furthermore, many solutions utilize special hardware to achieve the real-time behavior, which is expensive and usually proprietary. This results in strong constraints in the planning and deployment phase of a network. PROFINET IO/IRT requires special switches, as the network frames are forwarded on a fixed route that is determined by the transmission time. In [11], a real-time Ethernet approach based on the master-slave principle is developed where any device in the network can act as master. In order to achieve real-time behavior, the data link layer is exchanged with a custom protocol. This results in the necessity of special hardware and incompatibility with common Ethernet. [12] also proposes a real-time capable Ethernet protocol adopting the master-slave pattern. Here, the master represents a SPoF and the slave nodes require special hardware to participate in the network. Santos et la. present a scheduling framework, where a hierarchical server architecture is used to enable a dynamic bandwidth reservation for certain message streams [13]. As prove of concept, the approach was implemented in specialized switches as hardware software co-design. This concept is called Hard Real-Time Ethernet Switching Architecture (HaRTES). In [14], HaRTES is extended with the capability of multi-hop communication over multiple HaRTES switches. To achieve this, a distributed scheduling algorithm called DGS is presented. However, the utilization of specialized switches degrades the interoperability and results in higher deployment costs. Schmidt et al. present a real-time Ethernet solution without the disadvantage of a central instance called DRTP [15]. In the developed concept, two proprietary layers are introduced directly above the Ethernet layer to control the medium access. Hereby, the medium access scheme is based on a TDMA approach with defined time slots. However, the missing support of TCP/UDP and IP prohibits a total vertical and horizontal integration. The hard real-time P2P approach HaRTKad by Skodzik et al. also employs a TDMA approach to manage the medium access. But in contrast to [15], it is based on unchanged versions of Ethernet and IP/UDP.

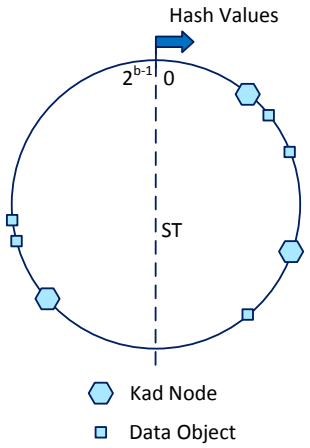

Fig. 1. Structure of Kad address space illustrated as a ring including nodes, data objects and the search tolerance ST.

III. HaRTKad

HaRTKad is a hard real-time capable modification of the distributed hash table (DHT)-network Kad, which is a specific implementation of the Kademlia protocol. Kad realizes a fully decentralized structured network where all nodes and data objects are identified through a unique ID, also referred to as hash value. The hash value is calculated through a hash function, like Message Digest 4 or 5, from a fixed and unique node name or a node's IP or MAC address. Every node in a Kad network is responsible for a certain set of data. This data set is specified through the search tolerance ST. If the distance D between the node's hash value and the hash value of the data is smaller than ST the node is responsible for the data. In order to determine the distance between two hash values, the XOR metric is used as described in Eq. (1).

$$D = Hash_{node} \oplus Hash_{data} < ST \tag{1}$$

The address space in a Kad system can be illustrated as a ring, where nodes and data objects are placed according to their hash value as depicted in Fig. 1.

The ST divides the address space in different areas of responsibility that are represented by the dark blue dotted lines. In order to retrieve data from another node, a Kad node must at first find the Kad node responsible for the data object. Therefore, each node maintains a routing table. This routing table contains information of many nodes with a short and only few nodes with a long distance to the own hash value. When a node (node A) is searching for a data object, it first contacts the node with the lowest distance to the data object (node B). If node B is responsible for the requested data, it will answer with the data. Otherwise, it will return the contents of its routing table. Then, node A contacts the node with the lowest distance to the data object from node B's routing table. This is repeated until a maximum number of search steps is reached (timeout) or a node responsible for the data is found. While the worst case complexity of a node search is $O(n)$,

the average complexity is $O(log_2(n))$, where n is the number of network participants. The low average complexity results from the special structure of the routing table and the fact that the Kad nodes and the data object share one address space. To fulfill the hard real-time requirements of industrial applications, information must be exchanged in a deterministic manner. Industrial Ethernet solutions achieve this through an arbitrary media access controlled by a central instance. This central instance would, for example, assign certain time slots to the network participants when a TDMA-based approach is applied. However, there is no central instance in a Kad-network to manage the network access. Hence, a node must know by itself when it is allowed to communicate over the network. HaRTKad uses the hash value of a node as basis to determine a node's time slot. This correlation is possible because both, hash value and time slot, are unique. In order to calculate the time slots for each node, HaRTKad utilizes the search tolerance. Kad originally uses a static value for the search tolerance, which is not sufficient for the purpose of time slot calculation. Therefore, HaRTKad adopts and modifies the algorithm presented in [16], where the ST is adjusted at runtime to ensure that at least one node is responsible for each possible data object. However, this algorithm requires knowledge about the number of network participants and their position on the hash ring. To gather this knowledge, [17] describes a mechanism that allows a Kad node to discover all other network participants with a high probability and without a-priori knowledge. This mechanim is called KaDis (Kad Discovery) and will be executed by a master node. The master is determined through a competition phase of all peers, where the first node to start the KaDis algorithm will be the master node. The role of the master is assigned to another node in the network if the initial master leaves the network. HaRTKad modifies the aforementioned algorithm for the dynamic ST adjustment, so that each area of responsibility contains a maximum of one node. This algorithm is referred to as inverse dynamic search tolerance (IDST). The newly obtained value for the search tolerance ST_{IDST} will be stored consistently by the IDST algorithm on all nodes. Fig. 2 shows an example where the hash value size is 4 Bit and three nodes are participating in the HaRTKad network. The IDST algorithm is executed periodically in a maintenance phase that is introduced in HaRTKad. In this way, joining or missing peers can be detected and removed from or added to the time cycle.

Based on its hash value and ST_{IDST} each node can determine its own time slot via Eq. (2).

$$Slot_{node} = \frac{Hash_{node}}{ST_{IDST}} \tag{2}$$

The total number of available time slots also depends on ST_{IDST} as described by Eq. (3). Here, b denotes the width of the hash values.

$$N_{slots} = \frac{2^b}{ST_{IDST}} \tag{3}$$

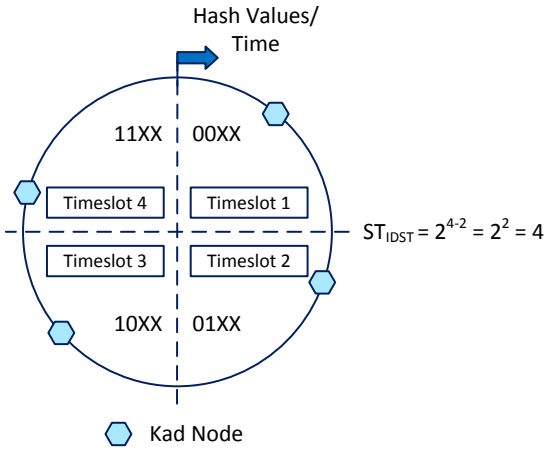

Fig. 2. Utilization of the search tolerance to determine the time slots for each node.

The cycle time, that can be seen as one turn around the ring, is calculated through Eq. (4). Here, t_{ex} stands for the time that is needed to find a node and exchange data with it.

$$T_{cycle} = t_{ex} * N_{slots} \tag{4}$$

Finally, the relative time in the cycle t_{ring} can be determined through Eq. (5).

$$t_{ring} = t_{now} \mod T_{cycle} \tag{5}$$

In order to realize a TDMA access scheme, all network participants must have a common time basis for the current time t_{now}. Therefore, HaRTKad implements the distributed time synchronization scheme described in [6] that is also executed in the maintenance phase. After the IDST algorithm has finished and all nodes are synchronized, each node can calculate its time slot and decide independently through Eq. (6) whether it is allowed to use the communication medium. Hereby, t_{ring} must be significantly smaller then a node's time slot end, so that a started network transaction can be finished within the node's own time slot. The period of the maintenance phase depends on the clock accuracy of the devices.

$$(t_{ring} > Slot_{node} * t_{ex}) \wedge (t_{ring} << Slot_{node} * t_{ex} + t_{ex}) \tag{6}$$

In order to give guarantees on the real-time behavior, all network participants need to run the HaRTKad software and consider only their own time slot for communication. Otherwise, the real-time behavior can be compromised due to concurrent network access and collisions during the communication. Practical evaluations of HaRTKad can be found in [5]–[7].

IV. Limitations of HaRTKad

HaRTKad is one of the few purely software-based approaches to real-time communication over Ethernet and the only one

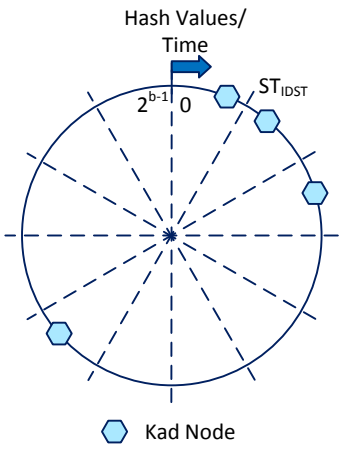

Fig. 3. Low medium utilization due to the high amount of created time slots.

among them to achieve isochronous real-time. Its placement on the application layer and the use of common Ethernet and the standard IP/UDP stack allow a total horizontal and vertical integration of devices. Still, HaRTKad suffers from some severe limitations. The first major problem is the possibly low network utilization in cases where nodes are clustered in a small area of the available logical address space. The low logical distance between the nodes leads to a very low value for ST_{IDST}. This may result in a high number of time slots of which only a small fraction will actually be used. Fig. 3 shows an example, where twelve time slots are created for only four nodes. Hence, in eight time slots the communication medium remains unused.

This problem can be solved through a balancing algorithm. [18] describes numerous balancing algorithms for DHT networks that establish a uniform distribution of nodes on the hash ring. Another major issue of HaRTKad is the handling of hash collisions that appear when two nodes have the same hash value. A feasible solution to this problem is described in section V. The third major weakness of HaRTKad is the missing ability to prioritize certain network traffic. Two possible approaches to solve this problem are discussed in section VI.

V. Handling hash collisions

When a new node enters the network, it generates its hash value. Although it is very unlikely, depending on the used hash function and length of the hash value, it can occur that the generated hash value is already assigned to another node in the network. This phenomenon is also referred to as hash collision. Under the assumption of an ideal hash function, where all hash values for random inputs are evenly distributed across address range, the probability $\boldsymbol{P}$ of hash collisions for n nodes in a b Bit wide address space can be calculated through Eq. 7.

$$\boldsymbol{P} = 1 - \frac{2^b - 1}{2^b} * \frac{2^b - 2}{2^b} * ... * \frac{2^b - (n-1)}{2^b} \tag{7}$$

Based on the Taylor expansion for e^x described in Eq. 8, an approximated value for $\boldsymbol{P}$ can be determined via Eq. 9. The approximation becomes more accurate for higher values of N and b.

$$e^x = \sum_{i=0}^{N} \frac{x^i}{i!} \tag{8}$$

$$\boldsymbol{P} = 1 - e^{\frac{-n(n-1)}{2^{b+1}}} \tag{9}$$

The probability for a hash collision for 100 nodes in a 64 Bit address space is approximately $2.68 * 10^{-16}$. Nevertheless, the occurrence of hash collisions can lead to several problems. First of all, two peers with the same hash value cannot find and hence, not communicate with each other. Furthermore, the real-time properties of HaRTKad can be compromised, as the two nodes would operate on the same time slot according to the IDST algorithm. This can put high pressure on the network infrastructure and lead to message loss and non-deterministic behavior. Therefore, it is necessary to handle hash collisions in a feasible manner. In the author's opinion, hash collisions should be already avoided when a node enters the network. Hereby, the entering peer calculates its hash value and searches for other peers in the network with the same hash value in advance. If a positive response is received, the entering peer needs to generate a new hash value and repeat the look-up procedure. The probability of multiple hash collisions in a row decreases exponentially with number of attempts, as the overall collision probability results from the multiplication of the individual probabilities for each attempt. In the given example with 100 nodes in a 64 Bit address space, the probability of hash collisions decreases to $7.2 * 10^{-32}$ for two subsequent attempts to calculate a unique hash value.

However, the proposed solution can lead to a falsely negative collision detection in cases where two nodes with the same hash value enter the network at the same time. Here, both peers will erroneously conclude that their hash value is unique. Although this scenario is rather unlikely to occur, it needs to be considered in order to give real-time guarantees. This issue can easily be solved in the HaRTKad maintenance phase, where each peer performs periodic self lookups and random lookups. Here, ST should be set to 1 so that a peer can lookup exactly its own hash value. All nodes, that have the concurrent peer in its routing table, will answer with its contact information. Typically, this information would be ignored, as it only reflects the peer's own contact information. However, this contact information contains the concurrent peer's IP address, so that it can be contacted directly. Afterwards, one of the peers must change its hash value and check again whether it is unique in the network as described earlier in this chapter. Which one of the two peers changes it's hash value is to be decided during the implementation process. Both peers entered the network at the same time, so that none of them is more established in the network. All other nodes will delete the peer with the changed hash value from their routing tables in the maintenance phase, as it is no longer responding to requests for the old hash value.

VI. Prioritizing Communication

Devices and applications in heterogeneous automation networks have different requirements regarding the timing behavior of the communication. The authors suggest to follow the categorization proposed by Mark Felser et al. and Gerhard Fettweis et al. [19] [20]:

- **Class 1**: 100 ms cycle time
- **Class 2**: 10 ms cycle time
- **Class 3**: 1 ms cycle time

Fully autonomous production lines require communication latencies of less then 1 ms, while other applications, such as the monitoring of slow changing processes, e.g. in a smelter, or consumer and multimedia applications tolerate communication latencies of up to 100 ms. HaRTKad does not take these different requirements into account. The time slots are assigned uniformly to all participating peers without prioritization of certain traffic. In conclusion, this means that peers with lower requirements have the same amount of communication time available as peers with higher requirements. This imbalance can be resolved in two different ways. The first solution would be to introduce flexibly scaled time slots. Originally, HaRTKad defines a fixed time slot length for all peers. In order to enable a prioritization of traffic, the time slot length for each peer could be scaled in regard of the real-time requirements. In consequence, the peers with higher requirements would have longer access times to the communication medium. Hereby, it is vital that the time slots are scaled in a feasible manner, so that the desired cycle time is not violated. However, it would be difficult for a node to determine the point in time it is allowed to access the medium. As the length of the time slot would differ for each peer, a node must be aware of the time slot length of every preceding node on the hash ring. The access criterion shown in Eq. (6) would change accordingly to Eq. (10). Here, t_{ex_n} is the time slot length of the nth node on the hash ring.

$$(t_{ring} > \sum_{n=0}^{Slot_{node}} t_{ex\_n}) \wedge (t_{ring} << \sum_{n=0}^{Slot_{node}+1} t_{ex\_n}) \quad (10)$$

The second and simpler solution is to allow the sharing of time slots among nodes. Hereby, multiple peers with lower real-time requirements share a single time slot and only access the medium in alternating cycles. This approach allows the prioritization of certain traffic without the modification of the basic time slotting mechanisms provided by HaRTKad. Furthermore, the maximum number of network participants can be increased if time slots are shared among several peers as depicted in Fig. 4. Here, all three class 1 nodes and the two class 2 nodes share a time slot, which allows three more class 3 nodes to enter the network without an increase in the number of time slots or modification of the time slot length. Theoretically, up to 99 class 1 nodes or nine class 2 nodes can share a single time slot when the cycle time T_{cycle} is 1 ms.

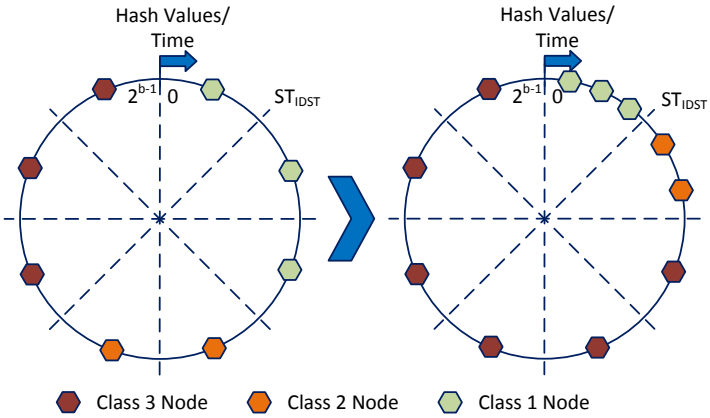

Fig. 4. Enhancement in the maximum number of network participants with fixed number of time slots and time slot length.

However, both approaches require information about the real-time category that is required by a node to decide which devices can share a time slot. In order to simplify the assignment of a time slot to multiple peers, it is desirable to cluster nodes with the same real-time class in the logical address space. The authors propose to dynamically split the address space according to the total number of peers and the number of nodes with the same time requirements. The KaDis algorithm can be easily modified to additionally gather information about the real-time class of each device on the master node. Thereafter, the master node can calculate the portions of the address space reserved for each real-time class as described by Eq. (11).

$$A_{class\#} = \frac{N_{class\#}}{N} * (s^b - 1) + A_{class(\#-1)} \quad (11)$$

Here, $N_{class\#}$ is the number of devices in the respective real-time class, N is the total number of devices in the network and b is the width of the hash value. Furthermore, $A_{class(\#-1)}$ is the address space occupied by the next lower real-time class. Afterwards, the values for the address space portions are distributed to all peers in the network alongside ST_{IDST}. Now, each peer can modify its own hash value to satisfy the criteria in Eq. (12) with regard to their own real-time class.

$$\begin{aligned} A_{class2} &> Hash_{class1} \\ A_{class3} > Hash_{class2} &\geq A_{class2} \\ Hash_{class3} &\geq A_{class3} \end{aligned} \quad (12)$$

Furthermore, the way ST_{IDST} is determined needs to be adapted so that a single time slot is assigned to multiple nodes. The simplest way to modify the calculation of ST_{IDST} is to only take nodes with high real-time requirements into account as depicted in Fig. 5.

The nodes that share a time slot communicate in alternating cycles in the order of their hash values. The sharing of time slots is to be preferred over the use of flexibly scaled time slots, as a node needs the ability to calculate its medium access

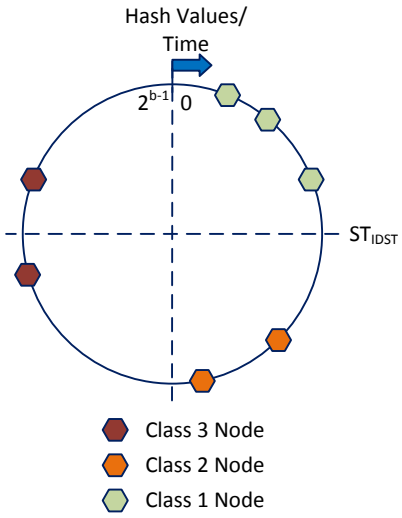

Fig. 5. Modified calculation of ST_{IDST}.

time on its own to avoid the necessity of a central instance. Even a combination of both approaches would inherit the disadvantages imposed by flexibly scaled time slots.

VII. Conclusion

In this paper, the handling of hash collisions, the low network utilization and the prioritization of certain traffic have been identified as the most significant limitations of HaRTKad. To solve the problem of hash collisions, a simple check-and-adjust approach that can be easily integrated into HaRTKad's maintenance phase has been proposed. Furthermore, two different approaches to allow traffic prioritization for nodes with higher real-time requirements have been discussed. Firstly, a flexible time slot length was suggested, so that nodes with higher real-time requirements would get more medium access time per cycle. The main drawback of this approach is that a node would no longer be able to determine the point in time when it can access the medium exclusively by itself. It either needs to know the time slot lengths of all preceding nodes on the hash ring or a central instance would be required to assign a time slot start and end to each node. Secondly, a more feasible solution, where multiple nodes with low real-time requirements share a single time slot, was proposed. This approach requires only minor changes in HaRTKad and will cause only negligible communication overhead. In our future work, HaRTKad will be extended with the proposed mechanisms. The feasibility of the described solutions will then be evaluated in a real world test bed consisting of up to 40 nodes and through simulation of larger networks.

Acknowledgment

This work was partially financed by the German Research Foundation (DFG) within the graduate school Multimodal Smart Appliance Ensembles for Mobile Applications (MuSAMA, GRK 1424).

References

[1] P. Danielis, J. Skodzik, V. Altmann, E. B. Schweissguth, F. Golatowsk, D. Timmermann, and J. Schacht, "Survey on Real-Time Communication Via Ethernet in Industrial Automation Environments," in *19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2014)*, Barcelona, Spain, September 2014, pp. 1–8.

[2] Panel Building & System Integration. Ethernet adoption in process automation to double by 2016, 2013. [Online]. Available: http://www.pbsionthenet.net/article/58823/Ethernet-adoption-in-process-automation-to-double-by-2016.aspx

[3] T. Sauter, "Integration Aspects in Automation - A Technology Survey," in *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, vol. 2. IEEE, 2005, pp. 9–pp.

[4] P. C. Evans and M. Annunziata, "Industrial Internet: Pushing the Boundaries of Minds and Machines," in *General Electric Tech. Report*, November 2012.

[5] J. Skodzik, P. Danielis, V. Altmann, and D. Timmermann, "HaRTKad: A Hard Real-Time Kademlia Approach," in *11th IEEE Consumer Communications & Networking Conference (CCNC)*, 2014, pp. 566–571.

[6] ——, "Time synchronization in the DHT-based P2P network Kad for real-time automation scenarios," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*. IEEE, 2013, pp. 1–6.

[7] V. A. B. K. E. B. S. F. G. D. T. Jan Skodzik, Peter Danielis, "CoHaRT: Deterministic Transmission of Large Data Amounts using CoAP and Kad," in *IEEE Internatilnal Conference on Industrial Technology 2015 (IEEE ICIT)*, Sevilla, Spain, March 2015.

[8] R. Pigan and M. Metter, *Automating with PROFINET: Industrial Communication Based on Industrial Ethernet*. Publicis Publishing, 2008.

[9] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluation of EtherCAT distributed clock performance," *Industrial Informatics, IEEE Transactions on*, vol. 8, no. 1, pp. 20–29, 2012.

[10] F. Klasen, V. Oestreich, and M. Volz, *Industrial Communication with Fieldbus and Ethernet*. VDE-Verlag, 2011.

[11] R. Schlesinger and A. Springer, "VABS - A new approach for Real Time Ethernet," in *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE*. IEEE, 2013, pp. 4506–4511.

[12] T. Hu, P. Li, C. Zhang, and R. Liu, "Design and application of a real-time industrial Ethernet protocol under Linux using RTAI," *International Journal of Computer Integrated Manufacturing*, vol. 26, no. 5, pp. 429–439, 2013.

[13] R. Santos, M. Behnam, T. Nolte, P. Pedreiras, and L. Almeida, "Multi-level hierarchical scheduling in ethernet switches," in *Proceedings of the ninth ACM international conference on Embedded software*. ACM, 2011, pp. 185–194.

[14] M. Ashjaei, P. Pedreiras, M. Behnam, R. J. Bril, L. Almeida, and T. Nolte, "Response time analysis of multi-hop HaRTES ethernet switch networks," in *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*. IEEE, 2014, pp. 1–10.

[15] K. W. Schmidt and E. G. Schmidt, "Distributed real-time protocols for industrial control systems: Framework and examples," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 10, pp. 1856–1866, 2012.

[16] P. Danielis, J. Skodzik, V. Altmann, L. Lender, and D. Timmermann, "Dynamic search tolerance at runtime for lookup determinism in the DHT-based P2P network Kad," in *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*, Jan 2015, pp. 355–360.

[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*. IEEE, 2011, pp. 1048–1055.

[18] P. Felber, P. Kropf, E. Schiller, and S. Serbu, "Survey on load balancing in peer-to-peer distributed hash tables," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 473–492, 2014.

[19] M. Felser, "Real-Time Ethernet - Industry Prospective," in *Proceedings of the IEEE*, vol. 93, no. 6, June 2005, pp. 1118–1129.

[20] G. P. Fettweis, "The Tactile Internet: Applications and Challenges," *Vehicular Technology Magazine, IEEE*, vol. 9, Issue: 1, pp. 64–70, March 2014.