HaRTKad: A **Ha**rd **R**eal-**T**ime **Kad**emlia Approach

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*—The Internet of Things is becoming more and more relevant in industrial environments. As the industry itself has different requirements like (hard) real-time behavior for many scenarios, different solutions are needed to fulfill future challenges. Common Industrial Ethernet solution often leak scalability, flexibility, and robustness. Most realizations also require special hardware to guarantee a hard real-time behavior. Therefore, an approach is presented to realize a hard real-time network for automation scenarios using Peer-to-Peer (P2P) technology. Kad as implementation variant of the structured decentralized P2P protocol Kademlia has been chosen as base for the realization. As Kad is not suitable for hard real-time applications per se, changes of the protocol are necessary. Thus, Kad is extended by a TDMA-based mechanism. Additionally, to evaluate the performance an prototype is presented, which is realized on an embedded platform with a real-time operating system. Thereby, with the presented approach and a realized prototype it is possible to investigate the performance of a Kad network with hard real-time capabilities.**

I. Introduction

In the field of automation, fieldbuses are widely established. They allow a deterministic data exchange between devices and follow the requirements for hard real-time applications. However, fieldbuses are limited in address space, scalability, resilience, and interoperability. Therefore, the industry tried to use common Ethernet technology, which has technical and economical advantages in contrast to fieldbus realizations. This led to the development of Industrial Ethernet solutions, which are able to guarantee hard real-time constraints like fiedbuses do. However, there is no common solution available as there are even more existing Industrial Ethernet solutions on the market than fieldbuses [1]. Industrial Ethernet solutions allow for a total vertical and horizontal integration of a automation system from the field level up to company level [2]. The number of participants is usually much greater that in fieldbuses as it is only limited to MAC or IPv4/v6 addresses. Additionally, the network is not limited to any topology in contrast to fieldbuses, which often require line or ring structures. Another economical advantage is the common Ethernet technology, which is already established and makes the components cheaper in terms of development and production costs than proprietary fieldbus solutions. However, Industrial Solutions also have disadvantages. Usually they possess a central instance, which represents a single point of failure (SPoF) and bottleneck due to the realization of the system following the master-slave or server-client approach. Other realizations require dedicated and expensive hardware to realize the hard real-time behavior. Additionally, many solutions leak flexibility and need a dedicated instance for administrative tasks. These issues will becomes more relevant in the future. As mentioned in [3], the future for the industry will be more intelligent devices, which can act more dynamically. Facilities as one main area of application will consist of more devices still requiring real-time or even hard real-time behavior. So we think, the existing solutions will not fulfill the future challenges in terms of scalability, flexibility, and robustness.

Peer-to-Peer (P2P) networks instead offer an innovative alternative to the typical Client-Server or Master-Slave concepts used in Industrial Ethernet solutions. P2P runs in the application layer and thus there is no need for special hardware or modifications on the lower layers. As already proposed in [3], the devices like sensors/actors and other facility components become more intelligent and offer more resources.

Therefore, an P2P-based approach using Kad is presented to realize a decentralized network of devices, which allows for hard real-time applications. The main focus is on the high robustness of the network and the scalable administration of the network. The Kad protocol has been modified to enable an arbitrary media access, which is necessary to meet timing constraints given by real-time applications. Additionally, the results of a working hard real-time Kademlia (HaRTKad) node is presented, which is required to realize a complete consistent system. The main contributions are:

- Presentation of the modified Kad protocol.
- Performance analysis of a HaRTKad client.
- Analysis of the HaRTKad protocol performance.

The remainder of this paper is organized as follows: In Section II, the related work is presented. In the following Section III, the basics of Kad are shortly described and the main approach to realize the HaRTKad system is explained. Section IV describes one HaRTKad node and the realization as a prototype and its performance evaluation. In Section V, an optimization step is presented, which significantly increases the number of HaRTKad nodes acting in parallel and thus increases the information exchange and channel utilization without violating the hard real-time constraints. Finally, the paper concludes in Section VI.

II. Related Work

Many Industrial Ethernet solutions like Ethercat, Ethernet Powerlink, Profinet IO/IRT, SERCOS III, and CC-Link IE Field base on the master-slave or client-server approach and

therefore show the disadvantages of central instances. The central instance represents a SPoF and also a bottleneck. E.g., Ethercat uses a master to synchronize the participants and to control the information exchange. Furthermore, some realizations, like Ethercat, TCnet, TTEthernet, and Profinet IO/IRT require special hardware, which is expensive and usually proprietary, which results in further constraints for a planned network. Ethercat for example needs special hardware for the slaves to achieve hard real-time performance. However, the use of special hardware for the Ethercat slaves results in expensive and proprietary hardware. Profinet IO/IRT needs special switches for forwarding packages in the network, as a they need to follow a predefined path [4].

In [5], a real-time industrial Ethernet protocol is developed adopting a master-slave pattern. The master is developed under Linux using Real Time Application Interface and represents a SPoF. Moreover, the slaves base on universal FPGA and ARM chips so special hardware is necessary.

[6] proposes a real-time Ethernet solution, which does not possess a central instance. The concept bases on two additional proprietary layers on top of the Ethernet layer to manage the media access. The result is an TDMA-approach using time slots. There is no analyses about a high number of attendees and its behavior in large scale networks. Furthermore, as TCP/UDP and IP are not supported the total vertical and horizontal is not possible without further effort.

We also propose a TDMA-based approach but directly integrate it in the Kad protocol. Thereby, we use the unchanged stack of Ethernet and UDP/IP. There are no changes needed except for the application layer. Kad will be changed to act as an application and also as an optional middle-ware for further applications on top of it. No central instance is required to configure, control, or synchronize the Kad network, which results in a well scalable, flexible, and robust network for automation scenarios, which require hard real-time constraints.

III. Basics and design concept

The distributed hash table (DHT)-network Kad (an implementation variation of the Kademlia protocol) has been used to realize a fully decentralized structured network. Every node has an own unique ID, which is called hash value. This hash value is usually generated by a hash function, e.g., the Message Digest 4 or 5 algorithm [7]. Every node is responsible for a set of data or information. The responsibility is given by the search tolerance ST. Kad generates a hash value for data or information and determines the responsible node by determine the XOR distance D between the data/information hash value and the node hash value. If the distance is less than the ST this node is responsible for the data/information. Formula 1 shows this correlation that a node is responsible if D is smaller than ST.

$$D = Hash_{Data} \oplus Hash_{Node} < ST \tag{1}$$

So the hash value directly influences, which data or information a node has to store [8].

In Industrial Ethernet solutions, the problem of fulfilling hard real-time requirements is solved by ensuring a deterministic information exchange. This is realized by using an arbitrary media access. The control of media access is done via central instances. In a TDMA-based approach, the central instance would assign time slots to the participants.

Kad instead has no central instance, which could control the media access. Therefore, a node needs to know by itself if it can access the media. A node's hash value $Hash_{Node}$ is unique and serves as information to determine time slots, during which a node is allowed to access the media. Like the relationship between data/information and the hash value, we suggest to establish a correlation between access time and the hash value as both are unique.

To determine the network size in a previously unknown Kad network and to adjust the search tolerance ST so that for hash value at least one node in the network is responsible, the originally used static ST value is insufficient. Thus, the search tolerance must be *dynamically* adjusted at runtime if new nodes join or leave the Kad network or fail. By means of an already developed algorithm (called Kademlia Discovery (KaDis) algorithm [9]), the discovery of all nodes is possible with a high probability without requiring a node to have a priori knowledge of the network. Hence, any node can calculate the search tolerance depending on the nodes present in the network (i.e., their hash values) and the width of the address space after having discovered all nodes by means of the KaDis algorithm. In addition to the KaDis algorithm, which is executed periodically, the calculating node's routing table is regularly checked for failing nodes and if changes are observed, the search tolerance is recalculated. After the search tolerance calculation, the calculated value is sent to all nodes.

Contrary, this algorithm could be easily modified to guarantee that a maximum of one node is responsible for a any hash value. This new algorithm will be called inverse dynamic search tolerance (IDST). Figure 1 represents a DHT ring where the search tolerance ST_{IDST} has been determined with the IDST in such a way that maximum one node is responsible for any hash value. The new value for ST_{ISDT} will be stored consistently by the IDST algorithm on every node in the network.

A main requirement for a TDMA-based time slotted network is the common time base throughout all participating nodes. Therefore, all nodes of the Kad network need to be synchronized. An approach to synchronize a fully decentralized P2P network based on Kad without a central instance has been presented in [10]. The synchronization of a big network can be achieved very fast by using the concept of helping nodes. Additionally, the synchronization is realized as deterministic approach to make it suitable for hard real-time applications in automation scenarios.

After the IDST algorithm has been executed and the nodes are synchronized, every node knows when it is allowed to communicate (access the shared Ethernet communication medium). It is necessary to correlate the hash space of the Kad network with the time space to create the relationship between the hash values of the HaRTKad nodes and the time slots.

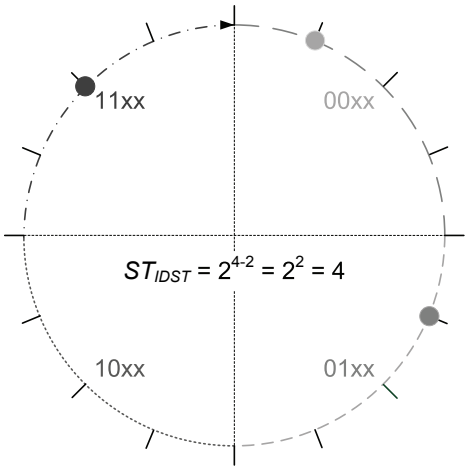

Fig. 1. Result of the IDST algorithm. The whole 4 bit address space is represented by a ring. Three nodes are exemplarily placed on the ring.

Therefore, an approach is presented to realizes the arbitrary media access of the nodes, which is handled in a decentralized way. Every node itself knows when it is allowed to access the media without a central instance as every node has sufficient information about the time slot it belongs to. The number of slots defined in Formula 2, which are generated, directly depends on the bit width of the hash values b and the ST_{IDST}. Additionally, every network participant must run the HaRTKad application as everybody must consider its own time slot to get access to the media. Otherwise, if a network participant is not part of the HaRTKad network it could compromise the network communication and the real-time behavior cannot be guaranteed.

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

The period T_{Del} represents the delivery time of a node, which can be seen as one turn around the hash ring. T_{Del} is given by Formula 3. t_{Ex} is the time needed to find an node and to exchange data.

$$T_{Del} = t_{Ex} * N_{Slots} \tag{3}$$

The actual time t_{Ring} based on the hash ring is simply expressed by Formula 4, where t_{Now} is the absolute time.

$$t_{Ring} = t_{Now} \mod T_{Del} \tag{4}$$

Now, it is important for a node to know, which slot $Slot_{Node}$ it is assigned to. Therefore, Formula 5 can be used.

$$Slot_{Node} = \frac{Hash_{Node}}{ST_{IDST}} \tag{5}$$

As each node knows the value for ST_{IDST} and t_{Now}, it can decide independently if it is allowed to sent. The decision criterion is represented by Formula 6.

$$(t_{Ring} > Slot_{Node} * t_{Ex}) \wedge (t_{Ring} << Slot_{Node} * t_{Ex} + t_{Ex}) \tag{6}$$

Using the presented formulas, we are able to realize a TDMA-based communication approach by correlating the time space with the hash space. As we are using the IDST algorithm, there is only one parameter left, which has to be determined. This parameter is t_{Ex}. t_{Ex} is a technical parameter, which is hard to determine theoretically as it depends on many parameters, especially on the used hardware. Thus, we have developed a prototype, which is presented in Section IV.

Another reason for not using the IP address of the nodes directly to assign a time slot is to support non-IP protocols on top of the Ethernet protocol as well. Therefore, we realize the assignment of the time slot in the application layer, so that we are able to support even proprietary protocol besides IP. Also, the IP address are not as equally distributed as the hash values of nodes generated by the MD4 or MD5 algorithm.

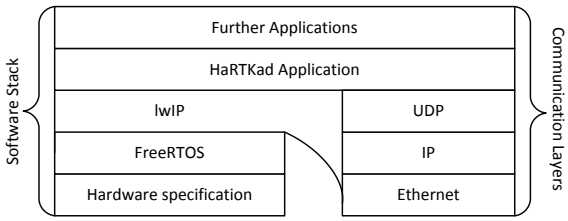

Fig. 2. Software stack of a Kad node

IV. Hard Real-time Kad node

As the presented approach is supposed to be used in automation scenarios, it is necessary not only to guarantee a deterministic packet exchange, i.e., communication among the nodes. It is also necessary to guarantee a hard real-time behavior of Kad nodes in terms of packet processing. Therefore, it is necessary to implement the Kad node in software using a real-time operating system. Hence, the target platform should support real-time operating systems. As target platform, the Zedboard with a 667 MHz ARM processor has been chosen [11]. We have chosen an embedded devices as we are targeting a platform for industrial automation. Using the developed HaRTKad prototype, it is possible to determine processing times of Kad operations and transmission times of exchanged UDP packets. The software stack and communication layer of the realized prototype are depicted in Figure 2.

Software stack: First, the hardware specification of the Zedboard has to be defined in software. However, only standard hardware has been chosen for the Zedboard consisting of the ARM CPU and RAM for software. It is possible to add own dedicated hardware components, which could improve the performance. In the first approach, we renounce the use of dedicated hardware to be able to realize the system with available standard hardware. FreeRTOS has been chosen as operating system to enable hard real-time behavior of the Kad nodes [12]. Additionally, lwIP is part of FreeRTOS as

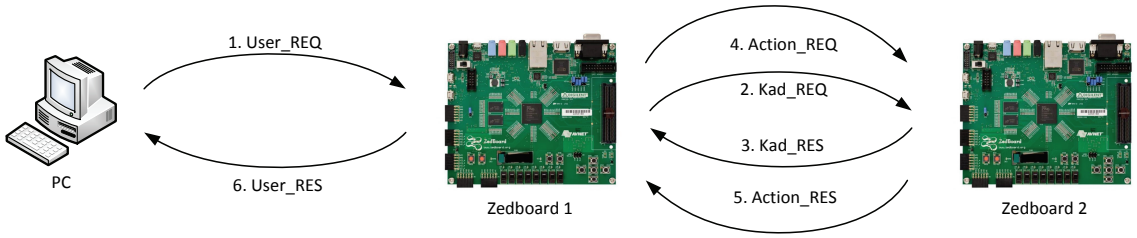

Fig. 3. Prototype setup for evaluation

a lightweight TCP/IP stack to enable the communication via the Ethernet medium [13]. The next layer is the HaRTKad application, which enables the real-time communication and the creation of the time slots for the nodes. Here, the presented approach is realized. Additionally, it is possible to use HaRTKad as middle-ware to run further applications on top of it.

Communication layer: The communication layer consists of standard Ethernet technology as the base medium. Also, the IP protocol is supported, which is necessary to enable addressing throughout the whole infrastructure. UDP as the next layers is used as standard TCP is not suitable for real-time applications. On top, we have the application layer, which consists of the HaRTKad application and an optional additional layer for another application where HaRTKad serves as the middle-ware.

The developed nodes support multi-threaded applications, which are needed by the Kad implementation. All Kad node threads are given in Table I with a short description. The threads are sorted by their priority starting with the highest one.

THREAD	PRIOR.	DESCRIPTION
Main	5	Starts other thread before going idle
External control	4	Receiving external commands
Kad communication	3	Processing of Kad packets
Search	3	Destroys search objects
Network	2	Network interface packet processing
Maintenance	1	Maintenance threads
Idle	OS	Origin of new threads

TABLE I
THREADS OF THE KAD CLIENT

The main thread is important to start the other threads and therefore got the highest priority. After having started the other threads, the main thread will stay idle. The external control thread got the second highest priority as it should react fast to external triggers like a human released fire alarm. The Kad communication thread has the next lower priority and is responsible for the processing of Kad packets. After the Kad communication thread, the three search threads handle the search in the Kad network. The network thread processes the packets from the network interface and forwards the packet data to the HaRTKad application. Three maintenance threads are responsible for keeping the network up to date. Finally, there is the idle thread, which is the source for new threads and it possesses a priority depending on the OS.

A. Performance Evaluation

The composite of the modified Kademlia protocol and the real-time Kad node allows to realize hard real-time applications based on P2P technology. It is necessary to build up a prototype scenario of HaRTKad nodes to evaluate their performance. Therefore, a prototype setting consisting of four main components has been realized to measure the performance values. One main computer, two Zedboards, and one 8-Port 1 GBit/s switch from Netgear [14] represent the prototype setup. The two Zedboards represent the Kad network and are logically connected via the Kad protocol. The setting is depicted in Figure 3. Two operations are supported in the setup: (1) read and (2) write operations between the two Zedboards, which are running the HaRTKad application.

(1) Read operation: If the user requests a read operation a number of integers will be transmitted. The number is given by the user and stored in the user request packet. Additionally, the hash ID of the nodes, which have to deliver the requested integer values, is given. The first Zedboard receives and processes this packet. As the first Zedboard is not responsible for the read request, it starts the search in the Kad network for a node, which his responsible for the given hash ID in the read request packet. Therefore, the first Zedboard contacts the second board via a Kad request packet to check if it is still available. The second Zedboard answers with a Kad response packet. When the first Zedboard receives the Kad response, it can contact the second Zedboard again as it is also responsible for the given hash ID in the read request packet by the user. This is done via the action request packets, which is the read action request in this case. The second Zedboard answers via the action response packet, which is the read action response packet. The read action response packet includes all integer values requested by the user and is filled with randomly created integer values. After the first Zedboard receives the read action response packet, it forwards the integer values to the user via the user response packet.

(2) Write operation: If a write operation is performed a given number of integer values is stored in the user request packet. Like during the previous described read operation, the corresponding node for the write operation will be searched in the Kad network. In this case, it is the second Zedboard as well. This receives an action request packet, which is a write action

request packet from the first Zedboard. This packet includes the integer values, which have to be stored on the responsible second Zedboard. The second Zedboard will send back a write action response packet as confirmation, which is forwarded by the first Zedboard to the user in form of the user response packet.

The times have been measured for all operations including the Kad operations and the processing of the packets. The first time measurement starts at the moment when a user request in form of a user request packet is received at the first Zedboard. The second time stamp is taken when the user request packet is sent back to the user PC. This has been measured for read and write processes. The result is depicted in Figure 4. As apparent, the results are below one microsecond. Additionally, it can be seen that with an linearly increasing number of integer values the time also increases linearly. These results can be considered for further inspection.

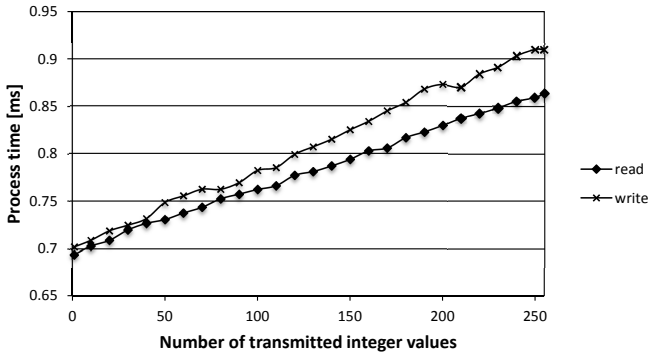

Fig. 4. Performance evaluation of a HaRTKad node

V. Optimization of the channel utilization

If using a synchronous TDMA-based approach time, slots could be unused. Therefore, one approach is presented to increase the channel utilization (CU).

Exploiting the processing time of a node: If the best case of the processing time of a node is known this time should be considered to achieve a higher CU. If we consider HaRTKad running on a Zedboard as target platform, it is possible to determine the CU during information exchange by Formula 7.

$$CU = \frac{Data_{packets}}{t_{Ex} * 1GBits/s} \tag{7}$$

It directly depends on the the data size, which has been exchanged. For the traced packets, we have six packets as previously described in the prototype scenario. The packets are summarized in Table II with their sizes, so that it is possible to compute the CU by computing the amount of data exchanged $Data_{packets}$. For further results, only the packets inside the Kad network are considered, the packets from and to the user PC are omitted. Additionally, to make the results comparable to one scenario, we assume that only one integer value (4 Bytes) is transmitted during the a read or write process.

Packet	Subtype	Size [Byte]
User_REQ		64
User_RES		64
Kad_REQ		89
Kad_RES		96
Action_REQ	read	88
	write	88
Action_RES	read	72
	write	72

TABLE II
Size of packets

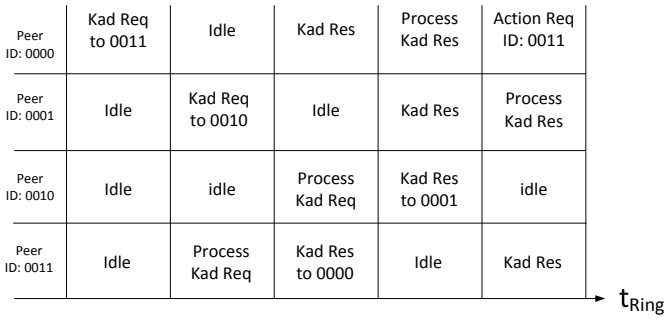

Fig. 5. Interleaving of media access to increase the CU

Most of the time, the nodes are processing packets and do not access the media. Thus, it is possible to allow parallel access to the media to increase the CU. This interleaving is exemplary depicted in Figure 5. The peer with the hash ID ”0000” contacts the peer with the hash ID ”0011”. Due to the time for processing the packets, the peer with the hash ID ”0001” is able to contact the peer with the hash ID ”0010”.

With the determined parameters, it is possible to indicate the number of nodes, which could theoretically access the media in one time slot while considering a theoretical channel utilization of 100 %.

In a nutshell, to indicate the performance of the presented network the work of Mark Felser in [15] is considered. Three classes are defined for human control, process control, and motion control. Furthermore, these three classes have different constraints in terms of timings. For a better comparability, it is assumed that an information exchange of 4 bytes between the nodes happen. The results from our prototype shows that the the finding of a node and the exchange of 4 bytes took about 700 μs, which already includes one search step of 150 μs. Every further search step took additional 150 μs, which is denoted as the time T_{Step}. Therefore, 550 μs are needed to send and process the action request and response packet and is denoted as T_{Action}. The number of search steps in the Kad network scales logarithmically with the total number of HaRTKad nodes as previously mentioned. It is also considered that we need more time to find other nodes in the Kad network, which is expressed by Formula 8. This is because if the optimum supports more node to achieve a higher CU, we need more time to search for nodes. If we chose the optimum number of nodes to achieve the highest CU, the increased time

needed to find a node is already considered.

$$Nodes = \left\lfloor \frac{T_{Del}}{T_{Action} + (log_2(Nodes) * T_{Step})} \right\rfloor \quad (8)$$

The solution of Formula 8 is given by Formula 9, where W is the Lambert W-function.

$$Nodes_{Max} = \left\lfloor \frac{T_{Del} * log(2)}{T_{Step} * W(\frac{2^{T_{Action}/T_{Step}} * T_{Del} * log(2)}{T_{Step}})} \right\rfloor \quad (9)$$

In Table III, the number of possible nodes is presented using the presented approach, only realized in the application layer and without any central instance.

If no optimization is realized we are only able to let a small number of nodes operate as prototypes. The amount of data, which includes the Kad packets and the action packets, is given to compute the CU. Additionally, it can be seen that the actual Ethernet media CU is very low. If we assume a theoretical CU of 100% it is possible to increase the number of nodes significantly.

ATTRIBUTE	HUMAN	PROCESS	MOTION
Delivery constraint for T_{Del} [ms]	100	10	1
No optimization			
$Nodes_{Max}$	68	9	1
t_{Ex} [us]	1600	1150	700
Data amount per T_{Del} [KByte]	84.34	6.28	0.34
CU [%]	0.69	0.60	0.38
With optimization			
$Nodes_{Max}$	5110	652	90
t_{Ex} [us]	2650	2200	1750
Data amount per T_{Del} [KByte]	12799.95	1279.80	127.88
CU [%]	99.99	99.98	99.90

TABLE III
SUMMARY OF THE HARTKAD SYSTEM PERFORMANCE

If we chose a human control scenario the CU is only 0.69 % if no optimization is applied. Due to the low CU, it is possible to increase the parallel communication. In a human control scenario, the number of nodes can be increased by the factor 75 from 68 to 5110. It is not possible to just increase the number by increasing the CU from 0.69 % to 100 %, which results in a factor of 144 because with the increasing number of nodes the number of search steps also increases, which results in a higher data amount per node to find another node in the Kad network.

To realize the optimization step it is only necessary to modify the IDST algorithm, which is described in Section III. The ISDT algorithm computes the ST_{ISDT} so that only one node is allowed to be active during a time slot. The number of slots N_{Slots} is kept and the number of nodes $Nodes_{Max}$ for the optimization is considered. The new approximated value for the nodes per slot $Nodes_{Slot\_Opt}$ for the optimization is given in Formula 10, which is close to the theoretical maximum as the hash values are well uniformly distributed on the hash ring.

$$Nodes_{Per\_Slot} = \frac{Nodes_{Max_{Opt}}}{Nodes_{Max_{No\_Opt}}} \quad (10)$$

The new $ST_{ISDT_{Opt}}$ is generated by the IDST algorithm by using the parameter $Nodes_{Per\_Slot}$.

VI. CONCLUSION AND FUTURE WORK

In this paper, an approach is presented to realize a fully decentralized Kad network meeting hard real-time constraints for connecting devices in automation scenarios. Additionally, the prototype for a hard real-time Kad node called HaRTKad node is presented.The combination of the presented modified Kad protocol and a working prototype allows for the realization of hard real-time applications with high resilience, flexibility, and without any SPoF. Furthermore, administrative scalability and usability in adding and removing further instances are simplified as no central managing instance is necessary. Every node is able to process given data in a deterministic way thereby ensuring hard real-time behavior. Prospectively, the presented approach will be used to evaluate an application in an practical scenario with a high number of HaRTKad nodes.

REFERENCES

[1] M. Felser, "Real Time Ethernet: Standardization and implementations," in *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, 2010, pp. 3766–3771.

[2] T. Sauter, "Integration aspects in automation - a technology survey," in *10th IEEE Conference on Emerging Technologies and Factory Automation, 2005. ETFA 2005.*, vol. 2, 2005, pp. 255–263.

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

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

[5] 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. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/0951192X.2012.731609

[6] K. Schmidt and E. 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.

[7] R. Brunner and E. Biersack, "A performance evaluation of the Kad-protocol," *Institut Eurecom, France*, 2006.

[8] R. Steinmetz and K. Wehrle, Eds., *Peer-to-Peer Systems and Applications*, ser. Lecture Notes in Computer Science, vol. 3485. Springer, 2005.

[9] 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 *36th IEEE LCN, 4th International Workshop on Architectures, Services and Applications for the Next Generation Internet, pp. 1060–1067*, 2011.

[10] J. Skodzik, P. Danielis, V. Altmann, and D. Timmermann, "Time Synchronization in the DHT-based P2P Network Kad for Real-Time Automation Scenarios," in *The Second IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services, IoT-SoS 2013*, 2013.

[11] Avnet. [Online]. Available: http://www.zedboard.org/

[12] Real Time Engineers Ltd. [Online]. Available: http://www.freertos.org/

[13] Free Software Foundation, Inc. [Online]. Available: http://savannah.nongnu.org/projects/lwip/

[14] Netgear. [Online]. Available: http://www.netgear.com/service-provider/products/switches/unmanaged-desktop-switches/GS108.aspx

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