Buffering Algorithms for Modular, Decentralized Controlled Material Handling Systems

S.Sohrt, Z. Seibold, T. Krühn, L.Prössdorf, L. Overmeyer, K. Furmans

ABSTRACT

In this paper we present two conceptually different methods for buffering packets on modular, decentralized controlled conveyors. At first, we give a short overview over current research projects. Since buffering is closely related to the storage of goods, we also address modular, decentralized controlled storage systems. We then show the general outline of the first algorithm which works with static assignment of buffer spaces. This algorithm assigns a value to every buffer space to assess its suitability. The suitability is defined by the likelihood of buffered packets interfering with the regular transport process. Packets are buffered on the most suitable buffer space and not moved until they get requested. The second algorithm is using dynamic assignment of buffer spaces. Thus, buffered packets can still be moved. We present the emerging behavior of this algorithm arising from the dynamic reassignment of buffer spaces. Both algorithms are designed in such a way that collisions and deadlocks cannot occur.

1 INTRODUCTION

Conveyors and storage systems face new challenges: Automated systems are too inflexible to adapt to the rapidly changing demands in structure, volume and processes of today's material handling processes [1, 2]. Therefore, systems have been developed that are highly automated, flexible and reusable [2, 3].This is achieved by having a modular design and a decentralized control.

A decentralized control that reconfigures itself is necessary to ensure quick response times for layout changes. Due to the large number of modules in a system, it is not possible to program an individual control for each element. The solution is a decentralized control for the conveyors modules, thus all modules running the same program without a central supervision. Since the control is distributed, the system becomes less prone to a complete breakdown: Failure of single modules does not necessarily affect the function of the whole system. [3, 4, 5]

In this paper we will show that decentralized controlled systems are capable of controlling the transport process while finding suitable buffer spaces at the same time. Buffering is different from storing packets, since both the number of packets and the storage time is much lower than for storing [6].

This paper is organized as follows: In section 2, we introduce research projects in the field of decentralized controlled material handling systems. The Cognitive Conveyor and GridSorter are presented in more detail as section 3 describes buffering algorithms for these systems. Two different approaches to assign buffer spaces are presented before we conclude in section 4.

2 MATERIAL HANDLING SYSTEMS WITH DECENTRALIZED CONTROL

In this paper we will only consider modular conveyors with a decentralized control that have proven the mechanical feasibility of their approach. One key difference between the different systems is the size of the modules.

The main challenge of all projects is achieve efficient and deadlock-free system behavior while keeping the decentralized control algorithms as simple as possible. It has been shown that modular conveyors with a decentralized control are capable of transporting goods without collisions or deadlocks. [3, 4, 7]

An example for decentralized controlled material handling systems is the FlexConveyor [3, 7] (see Figure 1), a flexible conveying system built out of multiple, identical modules that can be easily plugged and unplugged. One module has at least the size of one packet. It is optimized for the task of transporting goods in a user-defined layout. An item can be introduced into the system at any source and is transported to its specific destination. Complex transport tasks of multiple items with different sources and destinations can be performed thanks to the decentralized control and usage of alternative routes.

Figure 1: A conveying network of FlexConveyor modules [7]

GridStore [8] is a modular and decentralized controlled storage system. The described methods have been proven to work on the FlexConveyor platform. A very high storage density and a high throughput have been achieved, which previously have been considered as conflicting objectives.

KARIS PRO [9] combines the aspects of conveyors and autonomous guided vehicles. The vehicles cannot only perform the transport of single items but are designed to form two different functional clusters, as shown in Figure 2: As discontinuous cluster, KARIS vehicles connect to each other in order to transport huge items. As continuous cluster, several KARIS vehicles form a conveyor line to realize high throughput of goods.

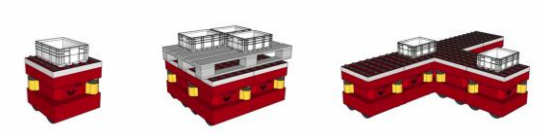

Figure 2: Single KARIS vehicle, discontinuous and continuous cluster

2.1 COGNITIVE CONVEYORS

The goal of the netkoPs research project is to develop a decentralized control for a modular conveyor matrix as shown in Figure 3. This matrix consists of modules that are smaller than the transported goods. It is capable of solving material handling tasks such as conveying, separating and merging [5].

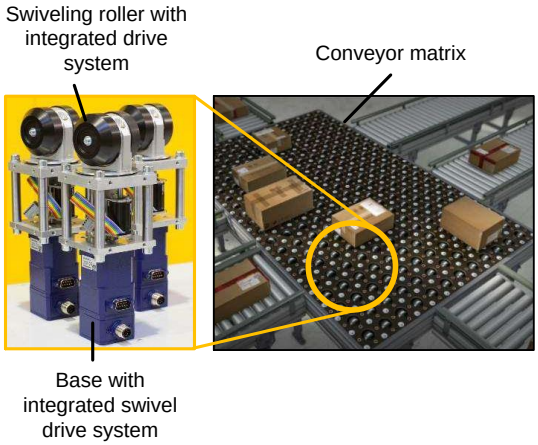

Figure 3: Prototype modules and the concept of a larger module matrix

The modules that form the matrix are not controlled by a higher-level, centralized control, but every module has its own control. Since each transport module is smaller than the packets, modules must form groups to conduct the transport process. When several modules are combined to form a conveyor matrix, it is mechanically able to solve any transportation tasks. Each material located on the matrix can be moved on an individual track. To transport a packet, a route reservation from source to sink is necessary. If multiple packets are moving on the matrix without a previous route reservation, a deadlock situation can occur.

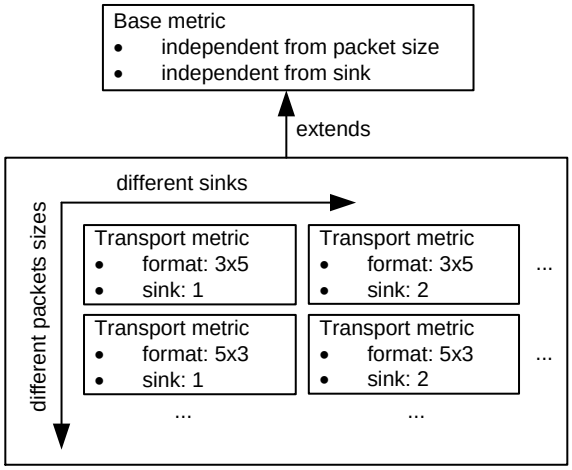

Figure 4: A new transport metric is generated for each sink and format.

To plan a route from one point in the matrix to another, the direction of the best route must be known in each module. Therefore, each module determines for all four directions the estimated virtual cost of moving a transportation unit in the respective direction. This procedure extends the distance-vector algorithm. Due to the two-dimensional

spreading of the metric and the small size of the modules relative to the packets size, the term field is used to describe the semi-continuous metric in this work. The metric is depicted in Figure 4.

2.2 GRIDSORTER

The GridSorter [11] consists of decentralized controlled conveying modules, for example FlexConveyors, forming a densely connected network (see Figure 5). Sources and sinks can be positioned at any position. In contrast to the conveyor matrix, the modules have the size of a packet. The basic functionality of GridSorter is the sorting of goods. An experimental setup with a 5x5 grid has been built in order to demonstrate the functionality (see Figure 6).

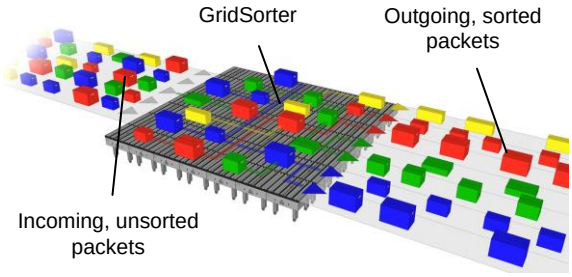

Figure 5 Schematic representation of GridSorter

There are three main processes that generate the system behavior: To detect the topology of the system every module registers with its direct neighbors. Hereupon, this connection information is propagated through the system. Each module establishes an adjacency matrix of the complete topology. With this matrix, the modules are able to compute necessary information about routes to possible destinations. The reservation process is started for every packet and defines the route of this individual packet to its destination. During the transport process, the modules coordinate the movement of the packet. Thanks to the modular design, further functionalities such as buffering and sequencing can be realized by enhancing the decentralized control algorithm.

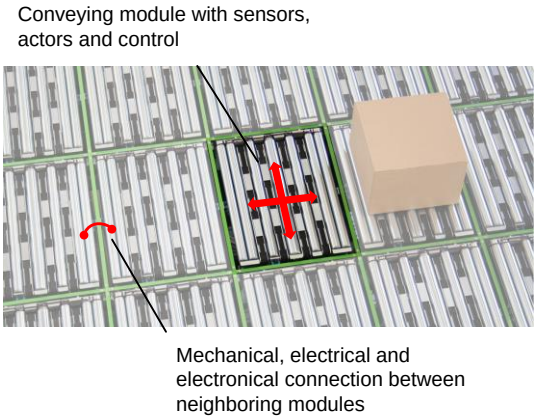

Figure 6: Technical realization of GridSorter: The conveying modules and their connections

3 BUFFER ALGORITHMS FOR DECENTRALIZED CONTROLLED CONVEYORS

In this chapter we present buffer algorithms for the Cognitive Conveyor and the GridSorter. The differences between the two approaches originate from two factors: First of all, the underlying routing is fundamentally different. The modules of the Cognitive Conveyor do not know the topology of the whole system, but only a very small local region. Routes are calculated and reserved by using a metric. Contrary, the neighboring information of each module of the GridSorter is propagated through the complete system. With this knowledge, the modules coordinate the route reservation using time windows.

Furthermore, different goals are pursued: Buffering on the Cognitive Conveyor should be energy efficient and thus movement of buffered packets is not permitted. The main objective of the buffering functionality of GridSorter is to reach a high buffering capacity. Both approaches to design the buffer algorithm are presented in the following two sections.

3.1 STATIC ASSIGNMENT OF BUFFER SPACES

The buffering of packets is done in two steps. First a suitable buffer space needs to be found. Afterwards, the packet is transported to this space. Since the system is already capable of transporting packets, the main problem is to find suitable buffer spaces. Once a space has been found, the buffer space is treated like a regular sink and a transport metric is created. There is one important distinction to make between sinks and buffer spaces: In contrast to regular sinks buffer spaces can overlap each other. Therefore, it is necessary that the

modules inform overlapping buffer spaces once they are in use. We define the following requirements for buffering on the matrix:

1. The buffering of packets must not lead to a partial or global deadlock of the system.

2. A criterion must be specified to rate buffer spaces.

3. Buffered packets should not be moved once buffered.

In order to meet the first requirement, the transport system needs to check if there is at least one buffer space that can be reached by a packet before entering the system. By having a criterion to rate buffer spaces, it is possible to use buffer spaces first that are better suited than others. The third requirement was formulated to prevent additional workload on the system that is created by moving an already buffered packet to another buffer space. Nonetheless the system is generally capable of moving already buffered packets to another buffer space: In this case the original buffer space acts as a regular source.

An additional metric is introduced for storing the suitability of buffer spaces. The new metric is called buffer metric (shown in Figure 7).

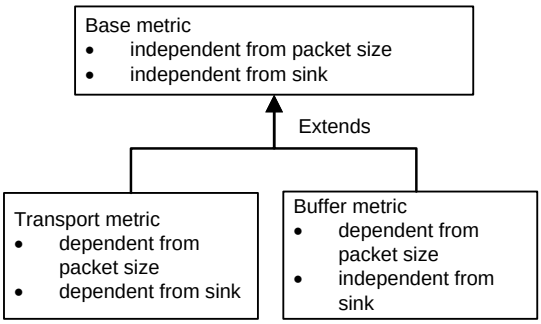

Figure 7: Relationship between the metrics

The buffer metric can be in three different states:

1. The module is permanently unsuitable as a buffer. A reason for this is that the module is too close to the edge of the matrix or that buffering a packet on this module would lead to a deadlock.

2. The module is currently unsuitable as a buffer. A reason for this that a buffer space is already occupied by a buffered packet.

3. The module is suitable as a buffer. In this case it is necessary to calculate a metric value that is used to rate the buffer space.

In the next chapter we will present a method to detect if a module is permanently unsuitable because of a possible deadlock scenario.

3.1.1 Deadlock Avoidance utilizing Reserved Buffer Spaces

To avoid deadlocks, the system needs to ensure that only packets can enter the system when a suitable buffer space is available. This is not a trivial problem, since communication in the system is not instantaneous: After a packet has been buffered on the last available buffer space, it takes some time until this information reaches every source. It is therefore necessary that sources only let packets onto the matrix, after they made sure that at least one suitable buffer space is available. This is guaranteed by the following method: Routes are planned from the source to the buffer space. The buffer space is then reserved and this information is sent back along the planned route to the source. After this is completed the source accepts the packet from the preceding system and transfers it onto the matrix.

Another possible deadlock situation can occur, when buffered packets fully or partially block bottlenecks. It is therefore necessary to only mark modules as possible buffer spaces if it is guaranteed that a buffered packet will not obstruct parts of the system. This is achieved by introducing an initialization phase: During this phase every source plans routes to every sink for the biggest known packet size. Every module that is needed to transport these packets from any source to any other sink is marked as permanently unsuitable for buffering. The algorithm to determine if a buffer space is suitable is shown in Figure 8.

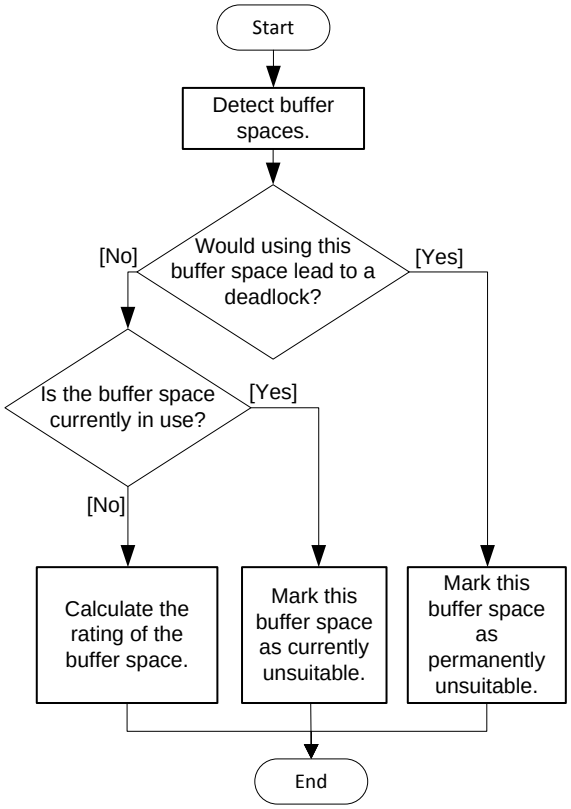

Figure 8: Algorithm to determine the suitability of a buffer space

After a packet has been buffered it is possible that it can be blocked by other buffered packet as shown in Figure 9.

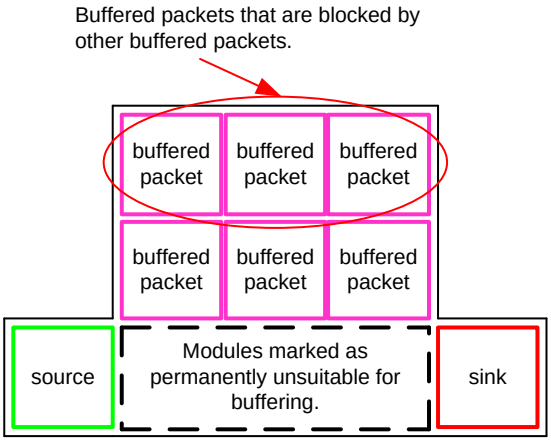

Figure 9: Buffered packets are blocked by other buffered packets.

The blockage by other buffered packets is avoided by guaranteeing that every buffered packet can reach every sink. This is achieved by reserving routes from the buffered packet to every sink. If a module is on such a reserved route, it is marked as currently unsuitable for buffering. To avoid race conditions between neighboring buffers paces that can block each other the routes must be reserved before the buffer space can be marked as reserved. The procedure is thus:

1. A preceding system wants to transfer a packet that needs to be buffered onto the matrix through a specific source.
2. The source plans a route to a buffer space.
3. The buffer space reserves routes to all known sinks.
4. The buffer space is reserved.
5. The confirmation of the reservation is sent back to the source.
6. The source transfers the packet onto the matrix.

After guaranteeing that a deadlock or a blockage of the system cannot occur due to buffered packets, it is necessary to define a criterion to rate the quality of the buffer spaces.

3.1.2 Criterion-based Choice of Buffer Spaces

The use of some buffer spaces interferes more with the regular transport process than others. By introducing a rating it is possible to differentiate between buffer spaces. Thus it is possible to use the better suited buffer spaces first. The chance of interfering with the regular transport process is increased for every free side of a buffer packet. The four possible cases are depicted in Figure 10.

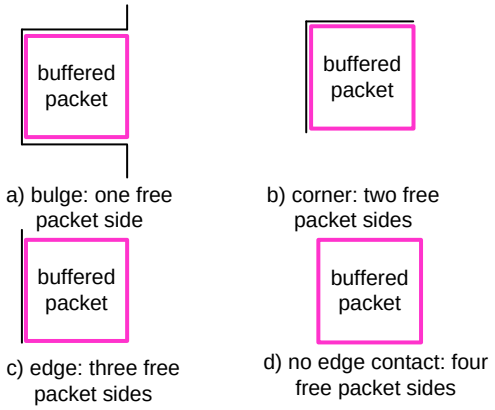

Figure 10: Four different buffer scenarios

It therefore makes sense to base the criterion to rate the buffer spaces on the distance of the sides of the space to the nearest walls.

3.2 DYNAMIC ASSIGNMENT OF BUFFER SPACES

The buffering algorithm is started once a packet without destination enters the system. Dynamic assignment of buffer spaces means that buffered packets do not need to stay on

the same buffer space once they are in the system. Contrariwise, they are moved from one buffer space to another as soon as they interfere with a planned route of another packet to its destination. In contrast to the buffering algorithm described in section 3.1, we do not use any additional criterion besides availability to assign buffer spaces.

We postulate the following requirements for our buffering algorithm:

1. The buffering of packets must not lead to a partial or global deadlock of the system.

2. We want to achieve efficient system behavior without using topology information to rate buffer spaces.

3. Buffered packets are allowed to move on the grid to clear the way for packets with destination.

The dynamic assignment of buffer spaces has the advantage that a high buffer capacity can be achieved. There are a lot of potential buffer spaces because the buffered packets do not permanently block routes. Theoretically, a system can buffer as many packets as the number of modules less two. One of the free modules is used to receive a packet with destination and the other free module is used to enable the movement of this packet [11]. Of course, there is a tradeoff between number of buffered packets and system throughput.

Because every module can temporarily be a buffer space, topology information is not necessary for the assignment of buffer spaces. With the dynamic assignment of buffer spaces, buffered packets are moved more frequently. This could lead – dependent of the specific buffering task – to higher energy consumption than with static assignment of buffer spaces. Also the reservation process gets more complex as the reservations of all buffered packets can interfere with each other.

3.2.1 Deadlock Avoidance utilizing Time Windows

As already mentioned, every module can be a buffer space. Thus, topology information does not impact the suitability. Nevertheless, a criterion has to be defined in order to guarantee deadlock-free behavior. The next section will introduces such a criterion.

The reservation process of GridSorter uses time windows meaning each conveying module on the chosen route to destination is only reserved for a specific time window. Thus, each module does know the currently last reservation after which it is available open-end.

A packet to be buffered must find a conveying module where it is allowed to stay infinitely. The suitability of the conveying module from a specific moment on is defined as follows:

1. No module is permanently unsuitable as a buffer as it is not necessary to keep free routes to every destination.

2. Modules can be temporarily unsuitable as a buffer space for the following reasons:

2a. The module is not suitable as buffer space from this moment on because the last reservation is later than this moment.

2b. The module is not suitable as buffer space because it is already planned to be a buffer space for another packet and thus reserved open-end.

3. The module is suitable as a buffer from this moment on.

Once a packet without destination is entering the system, a reservation process is started by sending the first reservation message. The module receiving the message checks its suitability and reacts according the following rules:

2a. If the module is unsuitable because the last reservation is later, the reservation message is forwarded to a neighboring module in a random direction.

2b. If the module is already used as buffer space, it accepts the reservation and sends a new reservation message. This is done to find a new buffer space for the packet that is currently occupying the module. Thus, the incoming packet pushes away the already buffered packet.

3. If the module is suitable, it confirms the reservation.

Not only packets to be buffered can push away other buffered packets, but also packets with a destination can push them away. Like this, we want to achieve that these packets reach their destination as fast as possible.

3.2.2 Emerging Choice of Buffer Spaces

As explained in the previous section, reservation messages for packets to be buffered are sent in random directions. No criterion is used to prefer any direction or to rate buffer spaces. We expect the following emerging behavior: Because a buffer space needs to be available open-end it is more likely that modules become buffer spaces that are less used by packets with destination.

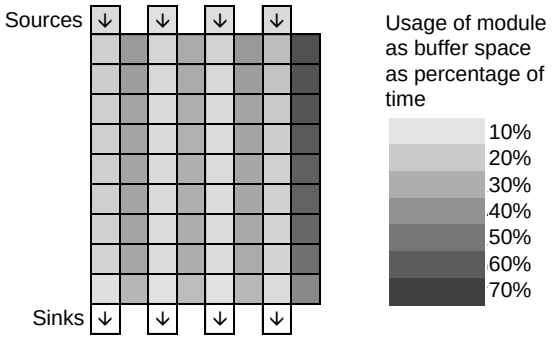

Figure 11: Usage of modules as buffer spaces over time with a coverage of 30% of the layout with buffered packets

Figure 11 shows a GridSorter layout with four sources in the north and four destinations in the south. The assignment of a destination to a packet is done randomly. Between these sources there are additional columns with modules and in the east there is a column that is not part of any shortest path from the sources to the destinations.

The layout has 72 conveying modules and 24 packets are buffered permanently. If a buffered packet is requested and leaves the system, a new packet to be buffered enters the system. This corresponds to 30% coverage of the layout with buffered boxes.

The gray value indicates the usage of the specific module as a buffer space over time. We can see that the columns between the sources and the destinations are less used as buffer space than the rest of the layout. The column in the east is used as a buffer space around 60% of the time. In this specific layout, the system shows the desired emerging behavior.

4 CONCLUSION

In this paper we presented two conceptually different buffering algorithms: One is designed for the Cognitive Conveyor and the other for the GridSorter. The main difference between both algorithms is whether movement of buffered packets is permitted once they have reached their buffer space.

The algorithm for the Cognitive Conveyor does not permit movement of buffered packets. Therefore, a rating is used to assess the suitability of possible buffer spaces. The suitability is defined by the likelihood of buffered packets interfering with the regular transport process. In contrary, the movement of packets is an integral part of the buffer algorithm for the GridSorter. Buffered packets are moved to dynamically assigned buffer spaces in order to clear the way for packets on the way to their destination. Movement of buffered packets has the advantage of high buffer capacity, but it could lead to higher energy consumption. Collisions and deadlock scenarios cannot occur in either of the algorithms.

We have shown that it is possible to realize high level logistic functionalities without a centralized control. Modular, decentralized controlled material handling systems offer the opportunity to accomplish complex intralogistics tasks with the same mechanical modules. We have developed buffer algorithms because they bridge the gap between storage and transportation.

We expect modular, decentralized controlled systems to become more mature. For this purpose different operational scenarios should be developed. It makes sense that one operational scenario is about the combination of all different systems. The next step is to analyze the influence of buffering on the throughput of the system. Furthermore, buffering is the base for realizing other high level functionalities like sequencing.

5 ACKNOWLEDGMENT

The research project netkoPs is supported by the “Bundesministerium für Bildung und Forschung“ of the German state. The research project GridSorter is supported by the „Bundesministerium für Wirtschaft und Energie aufgrund eines Beschlusses des Deutschen Bundestages“.

6 REFERENCES

[1] Cox, W. M.; Alm, R.: The Right Stuff: America’s Move to Mass Customization. In: Federal Reserve Bank of Dallas, Annual Report (1998), S. 3–26

[2] Furmans, K.; Schönung, F.; Gue, K. R.: Plug-and-Work Material Handling Systems. In: Progress in Material Handling Research (2010), S. 132–142

[3] Mayer, S.; Furmans, K.: Wissenschaftliche Berichte des Institutes für Fördertechnik und Logistiksysteme der Universität Karlsruhe (TH). Bd. 73: Development of a completely decentralized control system for modular continuous conveyors. Universitätsverlag Karlsruhe, 2009

[4] Krühn, T.; Radosavac, M.; Shchekutin, N.; Overmeyer, L. (2013): Decentralized and Dynamic Routing for a Cognitive Conveyor, International Conference on Advanced Intelligent Mechatronics (AIM), S. 436-441. Wollongong, Australia: IEEE/ASME

[5] Furmans, K., Gue, K. R., Seibold, Z.: Optimization of Failure Behavior of a Decentralized High-Density 2D Storage System, Dynamics in Logistics, Third International Conference LDIC 2012, Proceedings, 2012

[6] Gudehus, T.: Logistik – Grundlagen, Strategien, Anwendungen. Springer Verlage, Berlin / Heidelberg, 2010

[7] Mayer, S.; Furmans, K.: Deadlock prevention in a completely decentralized controlled materials flow systems. In: Logistics Research 2 (2010), S. 147–158

[8] Gue, K. R., Furmans, K., Seibold, Z., Uludağ, O.: GridStore: A Puzzle-Based Storage System With Decentralized Control, Automation Science and Engineering, IEEE Transactions on (Volume:11 , Issue: 2, pages: 429 - 438), ISSN: 1545-5955, 2014

[9] Furmans, K., Seibold, Z., Trenkle, A., Stoll, T.: Future requirements for small-scaled autonomous transportation systems in production environments, 7th International Scientific Symposium on Logistics, Proceeding, 2014.

[10] Ventz K., Hachicha M. B., Radosavac M., Krühn T., Overmeyer L.: Aufbau hochfunktionaler Intralogistik-Knoten mittels kleinskaliger Module als Cognitive Conveyor. Logistics Journal, Vol. 2012.

[11] Seibold, Z.; Stoll, T.; Furmans, K.: Layout-Optimized Sorting of Goods with Decentralized Controlled Conveying Modules. Systems Conference (SysCon), IEEE. 2013

[12] Gue, K.; Kim, B. S.: Puzzle-based storage systems. Naval Research Logistics (NRL) 54.5, 556-567. 2007.

Author:	Sohrt, Simon; Krühn, Tobias; Overmeyer, Ludger
University:	Leibniz Universität Hannover
Department:	Institute for Transport and Automation Technology
E-Mail:	{simon.sohrt; tobias.kruehn; ludger.overmeyer} @ita.uni-hannover.de
Author:	Seibold, Zäzilia; Prössdorf, Lisa; Furmans, Kai
University:	Karlsruhe Institute of Technology
Department:	Institute for Material Handling and Logistics
E-Mail:	{zaezilia.seibold; kai.furmans} @kit.edu