Wissenschaftliche Berichte des
Institutes für Fördertechnik und Logistiksysteme
der Universität Karlsruhe (TH)   Band 73

Prof. Dr.-Ing. Kai Furmans (Hrsg.)

Stephan H. Mayer

Development of a completely decentralized control system for modular continuous conveyors

universitätsverlag karlsruhe

Stephan H. Mayer

Development of a completely decentralized control system for modular continuous conveyor systems

Wissenschaftliche Berichte des
Institutes für Fördertechnik und Logistiksysteme
der Universität Karlsruhe (TH)
Band 73

Development of a completely decentralized control system for modular continuous conveyor systems

by
Stephan H. Mayer

universitätsverlag karlsruhe

Dissertation, Universität Karlsruhe (TH)
Fakultät für Maschinenbau, 2009
Referenten: Prof. Dr.-Ing. Kai Furmans, Prof. Kevin R. Gue Ph.D.

Impressum

Universitätsverlag Karlsruhe
c/o Universitätsbibliothek
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

Universitätsverlag Karlsruhe 2009
Print on Demand

ISSN: 0171-2772
ISBN: 978-3-86644-366-2

Preface

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Fördertechnik und Logistiksysteme der Universität Karlsruhe (TH).

Herrn Prof. Dr.-Ing. Kai Furmans danke ich für die wertvollen Ratschläge, sowie die Übernahme des Hauptreferats. Vor allem bei der feinen Unterscheidung zwischen wissenschaftlichem Arbeiten und reiner Entwicklungstätigkeit half er mir stets, den richtigen Weg zu finden.
I would like to thank Prof. Kevin Gue, Department of Industrial and Systems Engineering, Auburn University for his interest in my work and the honour to have him as second reviewer.

Der Bundesvereinigung Logistik (BVL) und der Arbeitsgemeinschaft industrieller Forschungsvereinigung (AIF) möchte ich für die finanzielle Förderung des Forschungsvorhabens danken.
Danken möchte ich auch meinen Kollegen, den Mitarbeitern der Institutswerkstatt sowie Hilfsassistenten und Studenten, die zum Gelingen der Arbeit beigetragen haben.

Auch möchte ich das Bundesverkehrsministerium nicht unerwähnt lassen, welches mir durch den Ausbau der A8 während meiner Promotionszeit viele inhaltsfreie, ungestörte Stunden im Stau schenkte, während dessen ich einen Großteil der Ideen zum Gelingen dieser Arbeit reifen lassen konnte.

Mein ganz persönlicher Dank gilt meiner Ehefrau Cindy Mayer für Ihre Unterstützung und die Geduld an einigen Wochenenden, an denen ich in Gedanken mehr bei der Wissenschaft als anderswo war. Mein Dank gilt auch meinen Eltern, die mir durch ihre Erziehung und die mir ermöglichte Ausbildung die Grundlagen mitgegeben haben.

Karlsruhe, April 2009 Stephan Mayer

Kurzfassung

Stephan Mayer

Entwicklung einer vollständig dezentralen Steuerung für modulare Stetigförderer

Um die Flexibilität und den Einsatzbereich von Stetigförderanlagen zu erhöhen, wird in der vorliegenden Arbeit eine vollständig dezentrale Steuerung für ein modulares Stetigfördersystem vorgestellt, welches Fördereinheiten wie beispielsweise Kleinladungsträger ohne jegliche zentrale Infrastruktur befördern kann. Basierend auf existierenden Methoden zum dezentralen Datentransport in IT-Netzwerken agieren die einzelnen Module autonom und koppeln sich nach der Positionierung zur benötigten Topologie selbständig zum funktionierenden Fördersystem zusammen.

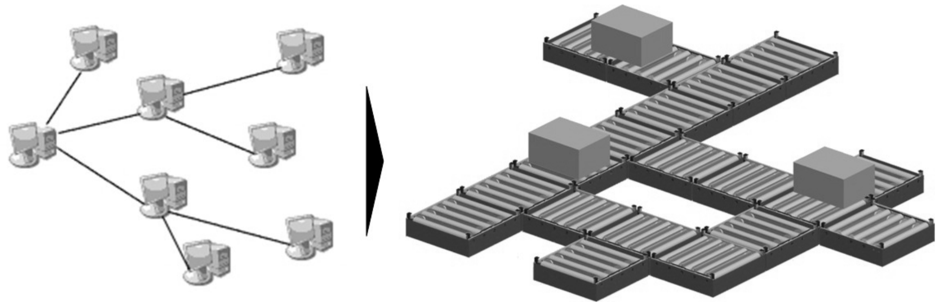


Figure 0.1: Übertragung der Methoden aus IT-Netzwerken für ein dezentrales, modulares Stetigfördersystem

Parallel zur Entwicklung der dezentralen Steuerung wurden baugleiche, quadratische Module entwickelt, welche als kompakte Einheit sämtliche Funktionen besitzen, um als Verzweigung, Zusammenführung oder einfache Förderstrecke zu fungieren. Dafür wird jedes Modul mit einem RFID-Identifikationssystem, Sensoren für die Positionserkennung der Fördereinheiten, einem Förderantrieb zum Transport in vier horizontale Bewe-

gungsrichtungen und einer Recheneinheit, welche den Steuerungsalgorithmus ausführt, ausgestattet.
Folgende Funktionen können die Module mit Hilfe des neuartigen Steuerungsalgorithmus ausführen:

- Selbständige Erzeugung der Topologielandkarte in Form von Routingtabellen
- Erkennung einer ankommenden Fördereinheit und Identifikation der Zieladresse
- Planung der Route bis zum Ziel unter Berücksichtigung der bereits im System befindlichen Fördereinheiten
- Absicherung gegen Kollisionen und Deadlocks und Transport der Fördereinheit zum nächsten Modul
- Selbständige Regulierung der Einlastung von Fördereinheiten, für einen höchst möglichen Durchsatz

Der entwickelte Steuerungsalgorithmus wurde in einer Simulation für repräsentative Topologien auf seine Durchsatzleistung untersucht. Weiterhin wurde ein Nachweis erbracht, dass unter bestimmten Bedingungen trotz Nutzung der Förderstrecken in mehrere Richtungen, niemals eine Situation entstehen kann, in der sich Fördereinheiten gegenseitig blockieren und es zum Stillstand des Materialflusses in Form eines Deadlocks kommt.

Abstract

Stephan Mayer

Development of a completely decentralized control system for modular continuous conveyors

To increase the flexibility and range of application of continuous conveyor systems, a completely decentralized control system for a modular conveyor system is introduced in the following dissertation. This system is able to carry conveyor units (for example, small load bearers) without any centralized infrastructure. Based on existing methods of decentralized data transfer in IT networks, single modules operate autonomously and, after being positioned into the required topology, independently connect together to become a functioning conveyor system.

Parallel to the development of the decentralized control system, identical square modules were developed, which in a compact unit contain all of the features necessary to function as a switch, junction or linear conveyor section. To fulfill this task, every module is equipped with an RFID (radio frequency identification) identification system, sensors for the position detection of conveyor units, a multi-directional drive to transport conveyor units in four horizontal directions, and a microcontroller-based control unit that

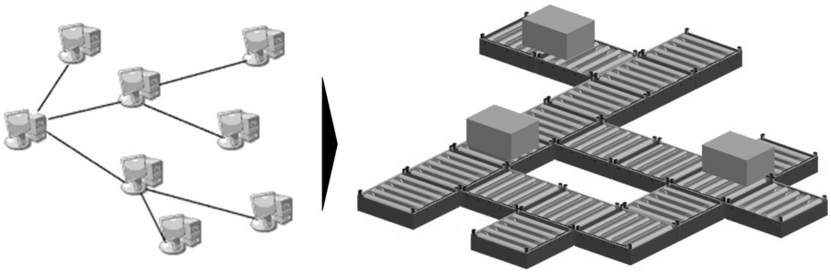


Figure 0.2: Decentralized methods of IT networks as basis for a decentralized, modular conveyor system

executes the control algorithm.
The following functions can be performed by these modules with the help of the innovative control algorithm:

- Independent generation of the topological map in the form of routing tables
- Recognition of an incoming conveyor unit and identification of the destination address
- Planning of the route to the destination taking into consideration conveyor units already located in the system
- Protection against collisions and deadlocks, and transportation of the conveyor unit to the next module
- Autonomous regulation of the injection rate to ensure the highest possible throughput

The throughput performance of the control algorithm developed here was analyzed by simulating representative topologies. Furthermore, it was proven that under certain conditions, despite the conveyor routes being used in multiple directions, a situation can never arise where conveyor units block each other and the flow of material comes to a halt in the form of a deadlock.

Contents

1 Introduction

1.1 Motivation

The internet and globalization lead to the fact that not only large corporations but also small and medium-sized corporations have to compete in the international market. The resulting changes in production have a direct effect on the processes of the in-plant flow of products and materials, and new demands on intralogistics emerge.

The biggest changes in the internal material flow are indicated by shorter project durations and declining batch sizes per order. E-commerce plays an enormous role by offering the possibility of a large variety of products in the shortest amount of time. Therefore, each company has to release new products to the market at an increasing rate in order to set itself apart from competing products. The simplification of the ordering process to a click has resulted in, among other things, a reduction of batch sizes. The customer orders more frequently and in smaller quantities. The delivered quantity per order is reduced to a minimum of one position and one piece, which results in a disproportionately higher amount of work for the supplier's order-picking systems. Therefore, the supply inventory is shifted from the buyer to the seller.

The following changes caused by e-commerce are to be considered (Seemüller 2006):

- Increasing rate of change in the variety of articles
- Volatile order batch size and rapidly fluctuating throughput demands
- Increasing volume due to more and more people ordering over the internet
- Fewer positions per order. This is a phenomenon which is caused especially by e-commerce
- Strongly fluctuating workload of the systems within a single day, over a period of several working days, as well as during the course of a year (e.g. Christmas business)

Because of these factors internal material flow systems have to become increasingly flexible. Inflexible systems like conventional conveyor belts are being replaced with more flexible systems, and instead of large scale systems, modular, scalable solutions are increasingly in demand. Existing flexible systems (forklifts, hand forklifts, picking carts, or hanging cranes) are uneconomical due to high labor costs and low throughput rates, and are only partially suitable for the transport of smaller conveyor units. To avoid the current compromise between flexibility and automation, plant manufacturers are increasingly trying to build modular plants that can be set up in the shortest amount of time in accordance with the demands of the customer. These modular components must still be installed and programmed individually, since no two conveyor systems are the same and the conveyor components cannot electronically integrate themselves automatically into the overall information system. A future change or expansion therefore requires a large investment, whereby a reutilization for other products or processes is not economically feasible.

Even if newer components already possess their own motor and sensor controls, the overall organization of the material flow elements is done centrally. The necessary programming effort negates the big advantage of modular systems, namely the simple, mechanical construction of the complete system by linking together a small range of standard components. Up to now this was necessary, because a centralized system architecture that controlled the system as a whole was used, which needed to be adapted to each individual layout at the time of installation or when changes were made. In spite of modular systems, long project development times, high costs for experts for the installation and modification, as well as long downtimes for repairs remain.

The continuous development of basic technologies has contributed to the fact that the prices of mass-produced items like CPU and memory chips, sensors, and identification systems are continually decreasing. This development allows thoughts about completely decentralized material flow systems where the individual components are equipped with all necessary electronics, thus allowing the organization of the material flow to be distributed among the components. Therefore, not only modular material flow systems can be developed, but also more autonomous and intelligent conveyor systems where each module takes over a part of the transportation task and exchanges information about the current system status with other modules. The immense increase in flexibility and the reduction of investment costs by avoiding the current programming and planning costs would especially enable small and medium-sized companies the increased use of automated systems. A change

of topology during operations would be just as easy as the replacement of single components in case of a malfunction.
Nevertheless, the biggest challenge of decentralized systems is the lack of an overview and with it the complex coordination of the conveyor units, so that no collisions or deadlocks can occur, especially during periods of heavy utilization.

1.2 Aim of the dissertation

The aim of this dissertation is to establish the foundation for the future application of completely decentralized material flow systems. At first, existing decentralized systems such as IT networks are analyzed and the structural differences in comparison to the transportation of physical conveyor units are established. An overview of existing approaches to the decentralized control of material flow additionally illuminates the latest knowledge and previously implemented concepts.
After defining the basic requirements and a clear range of functions, a new, completely decentralized control system for a continuous conveyor system is developed that distinguishes itself from all previous systems through a higher degree of autonomy and decentralization, as well as being able to operate completely independently from the infrastructure. Furthermore, all conveyor sections should be able to transport conveyor units in both directions.
Through the implementation of the control algorithms in a simulated environment, their performance is tested to determine their characteristics in relation to throughput and deadlock handling. The focus thereby is placed first and foremost on the stability of the system, so that in the worst case the flow rate will be reduced, but the system will remain functional at all times and a deadlock can never occur. Also, in the case of a technical failure of a module, the system should react appropriately. The knowledge gained should serve to prove the operational reliability.
The technical validation of the system occurs through the design and construction of several independent modules that are able to transport conveyor units, e.g., small load bearers (SLBs) or pallets, in accordance with the established basic requirements. At the same time, the control principle developed here is implemented in the modules and all functions are tested under laboratory conditions.

1.3 Structure of the dissertation

Chapter 2 "Centralized material flow control" gives an overview of conventional material flow controls. Following the classification of different types of controls, the most common hardware control - the programmable logic controller (PLC) - is discussed in detail and its mode of operation is examined. In conclusion, the limits of centralized material flow controls are identified to make clear the necessity to develop decentralized controls.

Chapter 3 "Decentralized control systems" discusses currently existing, decentralized control systems. To begin, a possible definition of the term *decentralization* is undertaken to be able to classify the data management as well as the data processing in material flow systems in relation to their *degree of decentralization*. Next, the operation of IT networks is illuminated, since there are several similarities between the transport of data packages and conveyor units that serve as the basis for the new control concept. The physical topology and the transmission of data packets are discussed in detail. Furthermore, the prevention of collisions and the creation of routing information using the two fundamentally different procedures of Distance Vector Routing and Link State Routing play a role. Finally, the physical differences between the transportation of data packets and conveyor units are discussed.
Chapter 3 ends with an overview of ongoing initiatives for the development of decentralized material flow controls by focusing on the "Internet of Things" and demonstrates the differences to this dissertation.

Chapter 4 "Completely decentralized, autonomous continuous conveyor system" describes my own approach to the development of a decentralized control system for transporting conveyor units without any centralized infrastructure. After delineating the problem and defining the basic requirements, the decentralized approach to a control system is described in detail. Four individual steps of the controlling process that fulfill the material handling task of a single module are introduced. After the manual connection to the conveyor system, the modules must independently generate the necessary information about the topology of the system. Furthermore, they must recognize an arriving conveyor unit and be able to identify its desired destination (sink). After routing and reserving the transport path, the conveyor unit must be sent on its way.
The focus thereby is on the avoidance of deadlocks, which can occur above all in complex layouts. After explaining the control algorithm, the origination of

a deadlock is addressed. It is shown how these deadlocks can be prevented, and also under which conditions a deadlock could still occur.
The last part of the chapter covers an analysis of the efficiency and capacity of the control system by investigating the throughput of various conveyor systems. The knowledge gained is ultimately used to develop a self-regulating mechanism that ensures that, even with a high injection rate of conveyor units, the system will consistently work at the highest possible filling rate.

Chapter 5 ”Technical implementation” introduces the first prototype of a module that is suitable for use in industry, and which, with the help of the control algorithm and upon being connected together with other modules, is capable of creating a material flow system with completely decentralized control. This chapter includes a discussion of the mechanical construction as well as the control components.

At the end of the dissertation, a summary of the gained knowledge and a short overview of further steps to the successful industrial application of a modular, decentralized control of material flow systems are given.

2 Centralized material flow controls

Until now, centralized control concepts have been used in material flow automation almost exclusively. Although there have been attempts to move more functionality closer to the actuators (e.g. the motor control mounted directly on the motor, which has an electrical power connection and a bus interface), the actual workflow logic still is in a central processing unit, which accesses the sensors and actuators via the bus system. However, it becomes clear that for the development of a decentralized material flow system, new demands on the peripheral hardware arise, which must be expanded to include the module for the workflow logic of the material flow.

In the following section, an overview of existing control methods is given. Afterward, the most commonly used control, the programmable logic controller (PLC), will be discussed in detail, in order to determine if existing hardware can meet the requirements of a potential decentralized application.

2.1 Classification of material flow controls

"Until today, control systems are used for the operative control of workflows. In the process, the relevant conditions are registered by sensors and transferred as input variables to the control system. The system interprets the input variables and forms output variables according to the given rules, which influence the real process with the assistance of actuators" (Jünemann and Beyer 1998).

2.1.1 Control concepts

Centralized controls

If the complete processing of the input variables of a system is done by an independent control unit that contains the complete logic, it is referred to as centralized control. Such a control unit could be a programmable logic controller (PLC) or an industrial PC. The capacity of the central unit is crucial for the entire system. Two identical units are often connected in parallel to increase availability. Such an arrangement can be used for applications that don't place very high demands on the main computer and where a malfunction won't cause excessive damage (see Figure 2.1a).

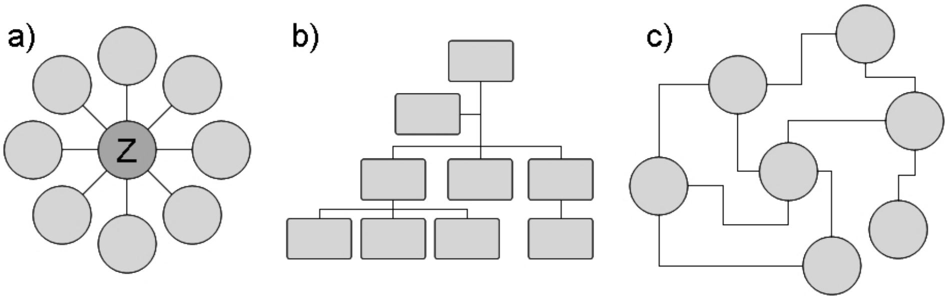


Figure 2.1: a) Central, b) hierarchical, and c) decentralized control concept

Hierarchical controls
Hierarchic controls have a clearly defined, firm concept of the division of labor. A central control unit distributes tasks to group control units, which in turn manage individual controls. This concept allows more demanding tasks than centralized control, because some of the calculations and decisions are processed on a lower level, thus relieving some of the load on the central computer. Consequently, a breakdown of the central unit does not always lead to an immediate breakdown of the entire system because the subordinate systems can fulfill their tasks on their own for a certain time.

Decentralized controls
Each control unit communicates on an equal basis with all others. At the same time, controls can be functionally subdivided according to specific abilities. Here, the capability of the individual components is not decisive, but rather the capability of the communication system. In the case of pronounced modularity, maintenance is simpler and the system can continue to be partially operative while being serviced (Haaß 1997), (Langmann 2003).

2.1.2 Structure of the control systems

Hardwired controls
The program flow of the hardwired programmable logic controller is defined by the hardware and is only changeable within a very narrow range without changing components. It is mostly realized electronically but it can also be done hydraulically or pneumatically. This was the only possible control before the invention and distribution of programmable logic controllers (PLC). Due to the enormous amount of time and effort required for the wiring and the very low flexibility, these controllers are only relevant today in special applications, for example, when maximum security (e.g. emergency-off circuits, security fence monitoring) or speed is required. But even here they are increasingly being replaced with PLCs (Jünemann and Beyer 1998).

Programmable logic controllers (PLC)
PLCs allow a fast and flexible change of the process cycle because the corresponding control logic exists as a software program that can be adapted quickly. During the physical reconstruction of the plant, further expenses arise, in addition to the programming, for the wiring of the input and output signals to the controller. Using bus systems can reduce this expense.

Industrial PC (IPC)
Due to the massive drop in price and the rising dependability of personal computers, they are increasingly being used as an alternative to PLCs for machine control. Therefore, the sensors and actuators have bus interfaces and are connected directly to the PC. The PC contains the control logic and communicates with the higher level systems. Because to date most sensors and actuators do not have an integrated bus interface, several sensors are connected together electrically in switch boxes to form a bus client.

2.1.3 Signal processing

Synchronous control
Signal processing is synchronized with the clock signal. The ports are read, processed and written to cyclically. Most PLCs are using this form of processing because, due to the structure of the computer, it is easy to implement. However, attention must be paid that the cycle time is short enough for the

application.

Asynchronous control
There is no clock signal. Each change in an input signal triggers a program sequence. This is realized, for example, with a relay control, which is not really very flexible because, being mechanical components, they can only be altered with a large amount of effort.

Logic control
With this method, the possible input signals are distinctly assigned to output signals using Boolean gates. The lack of storage elements is problematic, which is why a pure logic control is very rare.

Sequence control
The sequence control is a step-by-step procedure. The execution of the next step as required by the program depends on the stepping conditions. These can be conditional upon either time or a process. Time conditions, for example, allow a rest period for cranes to wait for the load to stop swinging. Process conditions use sensors to ensure that a load was correctly attached before it is set into motion.

These distinctive features cannot be assigned clearly to a certain control. Real controls are in fact a combination of the above-mentioned types as, for example, a synchronous logic control (Jünemann and Beyer 1998).

2.2 Tasks of the control levels in the material flow automation

In most cases, automated material flow systems are controlled at machine level with the support of a PLC. The problem here is logically connecting the data from the ERP (Enterprise Resource Planning) system, which controls the higher-level processes, to the PLC on the execution level. Current solutions are based on manufacturing execution systems (MES), which are located between the planning level and the execution level (see Figure 2.2). The EPR system plans the procedures and the MES carries them out. The MES processes the information from the ERP and generates the control signals. The communication occurs in detail as follows:

The MES receives transport orders from the ERP system and transforms them into orders with system coordinates for the controls level, because only these contain a detailed model of the layout. The system control, which can also be directly integrated into the MES converts the system coordinates into precise instructions for the PLC. Depending on the state of the conveyor system, the route of the conveyor units is optimized by the system control. Ultimately, confirmations of the successful completion of the operations or malfunctions are reported to the ERP system. New concepts allow the direct connection of the ERP system with the PLC through an extension of the PLC, whereby costs of central components are not incurred and decision authority is moved to a lower level (Arnold 2006), (Jünemann and Beyer 1998).

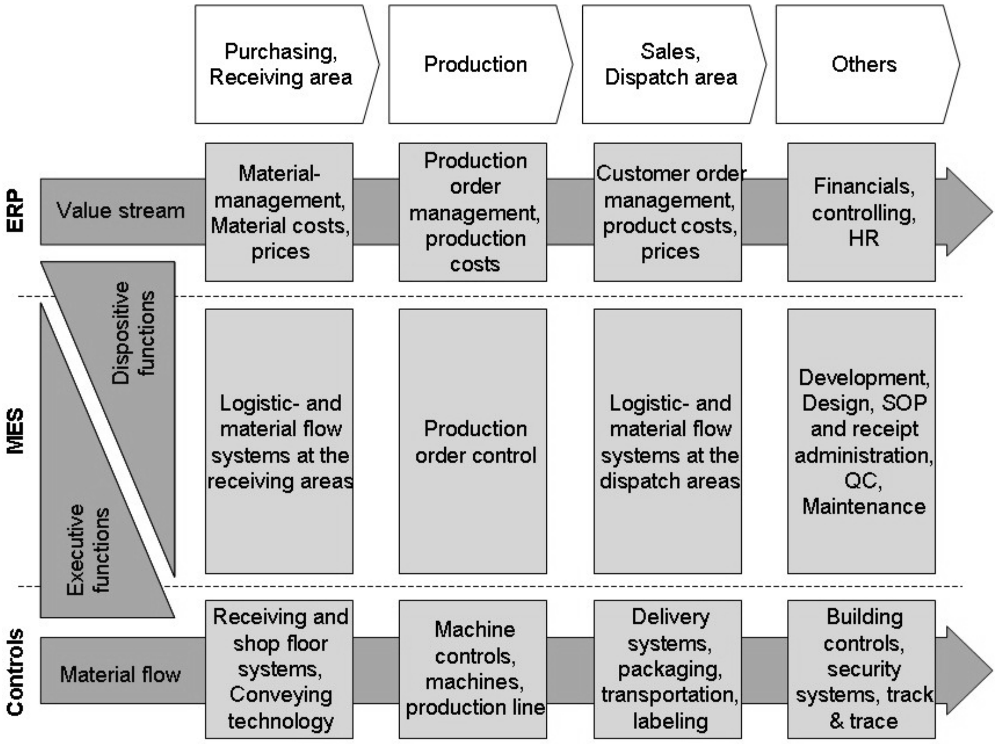

Purchasing, Receiving area
Production
Sales, Dispatch area
Others
ERP
Value stream
Material-management, Material costs, prices
Production order management, production costs
Customer order management, product costs, prices
Financials, controlling, HR
MES
Dispositive functions
Executive functions
Logistic- and material flow systems at the receiving areas
Production order control
Logistic- and material flow systems at the dispatch areas
Development, Design, SOP and receipt administration, QC, Maintenance
Controls
Material flow
Receiving and shop floor systems, Conveying technology
Machine controls, machines, production line
Delivery systems, packaging, transportation, labeling
Building controls, security systems, track & trace

Figure 2.2: Logistic core processes and IT levels, Source: Arnold 2006

2.3 Operating method of conventional PLC controls

The first programmable logic controllers (PLC) were invented in 1969. Since the 1980's they have been very widely used in the industry as controllers of machines and mechanical systems, and have replaced the hard-wired arrangement of the relay technology. Presently there are approximately 300 companies in europe that offer PLCs for various applications. The largest worldwide suppliers are Siemens, Mitsubishi, Bosch-Rexrodt, FANUC, and Rockwell Automation (Allen-Bradley).

2.3.1 Field of application of a PLC

The fields of application of PLCs are extremely diverse. From the control of roller shutters in house technology to the linking of machine tools to controlling and monitoring of large-scale chemical plants, appropriate PLC designs are offered. Figure 2.3 shows the typical construction of a Siemens PLC, designed to be installed in an in-plant electrical control cabinet. Distinguishing features are in particular the number of input and output channels, the cycle time, the performance of the CPU, and the expandability. A further selection criterion is the safety requirements placed on the controller (Wellenreuther and Zastrow 1998).

2.3.2 Structure of a PLC

A PLC fundamentally consists of a power supply unit, a central hardware unit that contains the RAM (memory), EPROM (erasable, programmable, read-only memory), ROM (operating system), and CPU (central processing unit), and the input/output modules (see Figure 2.4). The latter contain the connections for the wiring and the actuators, like motors or valves. A PLC can be expanded easily by adding additional input/output modules. Many individually-wired connections can be replaced by bus systems nowadays. With the help of analog-to-digital-converter (ADC) not only digital but also analog signals can be processed whereby the control of mechanical drives is made possible. To increase reliability, the central components of a PLC can

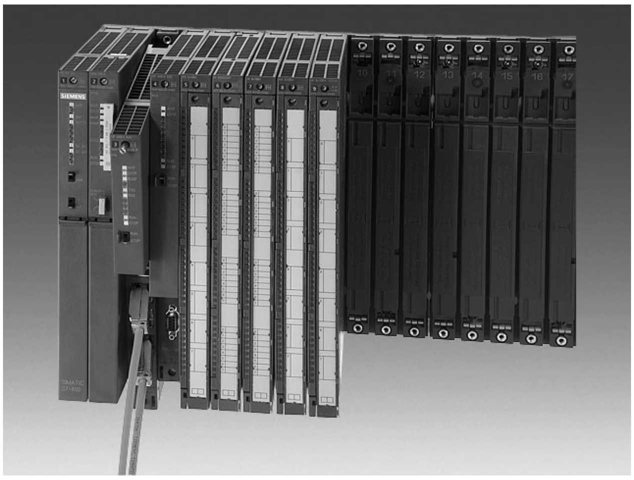


Figure 2.3: PLC (Simatic S7-400, Source: Siemens)

be integrated in parallel (Wellenreuther and Zastrow 2008),(Grötsch 2004).

	S7 300 CPU312-319	S7 400 CPU412-417
Bit operation processing time	0,1 μs - 0,2 μs	18 ns - 75 ns
Number of possible inputs/outputs	266 - 1024	32768 - 131072
Cost of central unit	380 - 530 eur	1000 - 11000 eur

Table 2.1: Performance data of current PLC central units

2.3.3 Programming languages

Several programming languages are generally available. The programmer can also switch back and forth between languages during programming. In general, there are no advantages or disadvantages. The choice of language often depends on the personal preference of the programmer. The ladder logic or ladder diagram (LD) language, which is in practice quite popular because of its graphic troubleshooting support, is symbolically and logically closely related to an electrical wiring diagram. Instruction list (IL) is very similar to Assembler and is considered the most powerful language because, as opposed to LD, it recognizes more commands and allows jumps. A compromise between IL and LD is the function block diagram (FBD), which understands

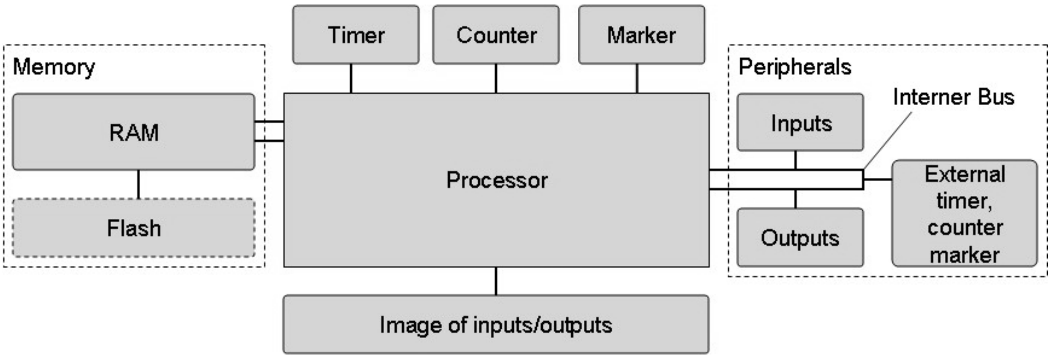


Figure 2.4: Internal structrue of a PLC, Source: Fern-Uni Hagen

more commands than LD, but the diagrams are more clearly arranged than in IL. Structured text leans more toward higher level languages like Pascal or C. The programs can be written, adapted and tested on PCs in the office and transferred to the controllers (Pickhardt 2000), (Rolle 1998).

2.3.4 Operating concepts

The operating concept of the PLC is of central importance. Most PLC units work **cyclically**, whereby the process image of all inputs is captured in a type of infinite loop. These input variables are processed according to the stored program into output variables, which are then set as the process image of all outputs. Typical cycle times lie between 1 and 50 ms.

In connection with cycle times, the subject of realtime data processing should be mentioned. The term realtime taken by itself reveals nothing about the system capability. Realtime guarantees only a deterministic time response in reaction to an input signal, meaning the adherence to particular time limits under all circumstances. These limits have to be established on the basis of the actual process. In the automation and material flow area they usually lie in the millisecond range. A further demand on realtime systems is the concurrent processing of multiple tasks (computing processes). For this parallel computers or scheduling procedures (usually priority controlled) are used (Jünemann and Beyer 1998).

Event-controlled concepts, where all input signals are processed in sequence, are becoming increasingly popular. The advantage here is the guaranteed processing of each signal. The length of time until processing is however not exactly predictable. At best, when no events are waiting, then the reaction to a signal can be immediate and the controller is faster than a cycle-oriented controller. If the scaling of central systems is too large, then even these controllers reach their limits because too many input signals must be processed in a short time (Grötsch 2004).

2.3.5 PLC networks

Networking of multiple controls is common these days. This makes it possible to transfer the system states and now allows the automated linking of machines with robots, loaders and material flow systems. A frequently used interface is the so-called Profibus. It is a specific type of bus system with defined standards so that components from various producers can be linked together.

Communication networks

To allow communication between two controls, both sides have a modulator and a demodulator. The modulator converts the signal to be transmitted, which is in the form of a bit pattern, into a physical signal. This signal can be optical or electrical. The physical signal is decoded in the demodulator of the receiver. Details of the transmission parameters are given in the communication protocol, like, for example, voltage level and the length of time for the representation of a bit. Next to this relatively easy transmitting assignment there are many supporting services. These primarily avoid disruptions caused by other signals that are on the transmission path at the same time through access control to the channel (e.g. token ring). The function of a token ring is briefly described here because the Profibus, which is described below, uses this procedure to control access to its participants. The token is the permit to transmit. It is handed-off after a pre-determined time interval among all active participants in accordance with a logical sequence. The bus member who currently owns the token is temporarily the master of the network and is able to send requests to the other participants (temporary slaves). They answer the master's requests but are not allowed to send any further data themselves.

The Ethernet, however, does not know this type of access control. The transmitter listens to the network and if there is no traffic, he transmits his data packet repeatedly until it returns to him correctly as an echo (Jünemann and

Beyer 1998).
The same functionality of packet shipping and collision avoidance is used as the foundation for the decentralized material flow control and is therefore described in more detail in chapter 3.3.2.

Profibus

The Profibus interface is standardized and allows communication between PLCs as well as between the PLC and actuators and sensors. Thanks to bus technology the necessary wiring is minimal. Profibus systems are primarily used in manufacturing cells and assembly facilities, where they take over the communication between machines from different manufacturers.
PLC processes are usually executed asynchronously. Thereby a high variability in the process and reaction time emerges. With the help of the equidistantly-working Profibus it is possible to read all inputs with chronological synchronism, to evaluate them and then to set the outputs with chronological synchronism. This guarantees consistency of the data and adherence to the logical order. The complete control process becomes deterministic. As a result a reduction of the control time is possible because the data from the inputs can be provided "just-in-time". This means the inputs are already being read ahead of a new cycle and are available to the program directly at the beginning of the cycle. The outputs are handled in the same manner, in that they are written to at the beginning of the next cycle. A time advantage at the beginning and end of each cycle is the result. However, the number of periphery inputs and outputs increases the cycle time. The longest time of the single components determines the total time of the system. The more components that are connected, the worse the time response becomes. Above all, the performance of the CPU is decisive (Wellenreuther and Zastrow 2008).

Industrial Ethernet

The Industrial Ethernet emerges through the integration of a networked control into a company-wide network, which, for example, can create the connection of a material flow management system with the underlying controllers. Also, the Industrial Ethernet can enable a direct access to shared machine parameters through the Internet. Figure 2.5 shows an example of a graphic user interface of such a remote diagnosis.
Further applications are remote diagnosis units that are integrated into mod-

ern systems and allow fast help with a malfunction of the controlled machines by the manufacturer (Metter 2007).

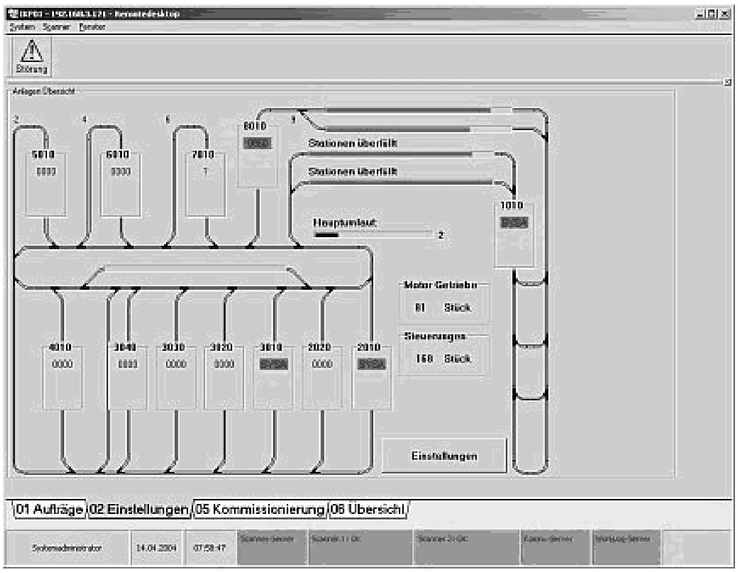


Figure 2.5: System diagnosis via the Industrial Ethernet

The capability of PLC processors and the size of memory modules are steadily increasing so that today, besides their main function, PLCs can also assume monitoring functions. The controls record the operating data of the machines and evaluate them. In addition to the graphical display on the human-machine interface (HMI) the data can be posted on the network and with suitable software tools, which are offered by many control manufacturers, made available to production planning and maintenance in realtime.

One concrete example of the Industrial Ethernet is the Profinet, which uses TCP/IP technology. The field bus integration describes the embedding of Profibuses in the Profinet. New kinds of transfer protocols (e.g., the IRT-mode) achieve cycle times that are shorter than 1 ms and therefore suitable for realtime applications. Motion control, i.e., the activation of actuators (e.g., CNC axes of a machine tool), is also integrated into the network. The connection of decentralized field devices (e.g., starter motors, input and output components) is handled in the same manner as are components of distributed intelligence using Profinet. An extensive security concept is available with Profisafe, which minimizes the loss of transmission data through, among other techniques, consecutive numbering of data packets, monitoring of the exact transmission time, and identification verification between the transmitter and the receiver. Remote diagnosis and web integration complete the functional spectrum (Pigan 2008), (Messerschmidt and Lüder 2002).

2.3.6 Limits of and alternatives to centralized controls for material flow systems

The PLC owes its prevalence to its stability and simple programming. For a long time PLCs were not allowed to be used in safety-critical areas because it was feared that software or hardware errors (e.g., integrated circuit diffusion) could endanger peoples' lives. Programs with built-in test functions, concepts like Profisafe and multi-channel systems alleviated the problem. Today, limits are primarily found in the areas of highly specific applications, for example, when extremely fast data processing and reactions to input signals are required. In these cases hard-wired systems are still used.
Because of the increasing flexibility requirements with regard to the adaptability of material flow systems, the programming effort involved with changes is moving more and more into the foreground, because a PLC always has to be readapted to the individual layout of the sensors and actuators. The problems thereby intensify when software updates are brought to the market in ever shorter time spans and it becomes more complicated to change the existing programming. Often it is hard to tell what impact a local change in hierarchical programming will have in other areas, because the complexity and the cross references become unmanageable in larger systems.
Another limit of conventional material flow systems lies in the restricted operational capability of existing systems. So far no conveyor system exists where all transport paths can be used in both transport directions. A change of transport direction is only possible with a large investment effort. A joint use of track sections in rail traffic, for example, are isolated, special-purpose solutions that until now have not been emulated in intralogistics.

Decentralization demands an increase in the functional range of individual components and the definition of new interfaces on a higher logical level. This simplifies the physical alteration of the transport system because necessary sensors and wiring remain consolidated with the material flow means. It is thereby necessary, however, that the control interfaces as well as the physical interfaces define a closed functional range and a ”plug-and-play” is created. This can be very well realized with existing PLC controls because processing power is sufficient for the additional logic for the coordination of the material flow. However, it could become necessary to install additional interfaces for this coordinating function. Alternatively, the PLC could also be replaced by small IPCs or simple computers, like, for example, microcontrollers. In this case, it must be ensured that the same programs are implemented on each individual, decentralized component and thereby reprogramming is avoided when the transport system is altered.

3 Decentralized control systems

3.1 Definition of the term "decentralized" in material flow

In the course of the increased flexibility demands of the industry on material flow systems, the efforts to develop decentralized material flow controls have increased sharply. The goal is to bring more and more decision-making competence closer to the actuators and also to implement related functions physically into a compact unit containing mechanics and electronics. These units can then be connected together with minimal installation effort to produce the desired material flow system (Günthner and Wilke 2002).

Several products have already been developed further following these requirements. Sensors now not only convert physical measurements into a voltage level but some also have a bus connection that sends the interpreted value to the logic component (e.g., PLC or IPC). Motors are increasingly being equipped directly with their corresponding control circuits so that to control the motor, only information about speed, ramp or end position are necessary (Messerschmidt and Lorentz 2002).

Although the products have a decentralized character, they are still far away from being able to fulfill complex conveying duties independently. Nevertheless, these systems are frequently described as "decentralized". On the other hand, various research projects are already working on giving conveyor units the ability to independently find their route from source to sink and to be able to call available conveyor means like a "taxi". Such a system would certainly have a higher level of decentralization than just a bus-connected motor (Bullinger and ten Hompel 2007).

Therefore, a definition and a classification of decentralization is introduced below that should help to classify the development initiatives and system solutions better.

Assignment of tasks
The control of a material flow system consists of various components that fulfill different tasks. Figure 3.1 shows the assignment of tasks on multiple levels on which varying information and logical cycles must be processed. The degree of decentralization of such a system is determined by the proportion of data storage and data processing that is situated locally, that is, near the sensors and actuators.

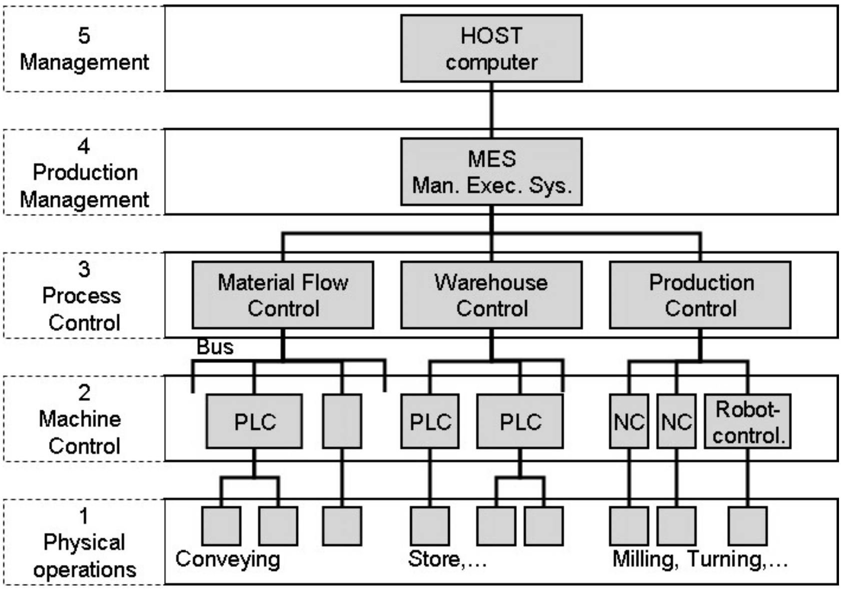


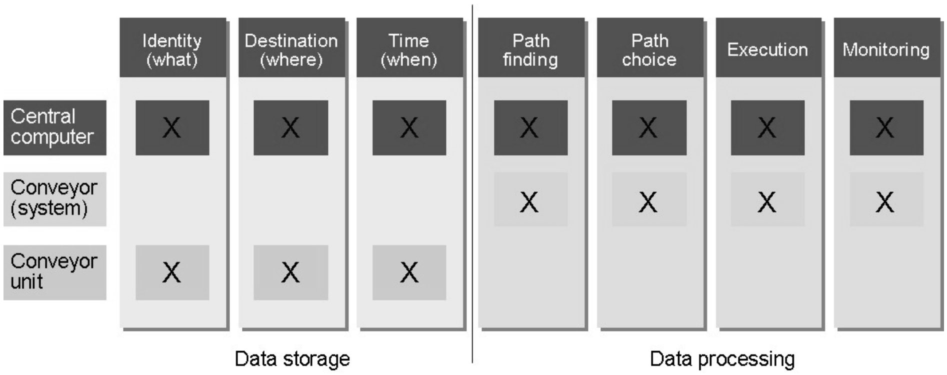
Figure 3.1: Distribution of control tasks in material flow systems on multiple levels, Source: Furmans and Arnold 2006

In modern intralogistic systems, instructions are generated by the central computer, which product **(what)** must have arrived at what point in time **(when)** at which destination **(where)**. Push systems generate these instructions ”downstream” along the production process via the central computer, whereas pull systems transmit these instructions ”upstream” from each local position to the upstream station. Consequently, pull controls work with a decentralized provision of information but the processing is mostly done as before centrally via the central computer, and only the triggering of an order occurs locally.
Therefore it is suggested to measure the degree of decentralization by de-

termining **who** owns and processes the three data of "what", "when", and "where" within the material flow control.
Additionally, which entity has which part of the work flow logic for the completion of the transporting task must be analyzed. The work flow logic is mainly occupied with the question of "**how**". This question can be answered with the following sub-processes:

- generating of the topology information (path finding)
- routing (path choice)
- actuator control (execution of the transport)
- supervision (monitoring)

Figure 3.2 shows the structure of data storage and processing, which can be handled centrally by the material flow computer, or locally by the conveyor, as well as by the conveyor unit itself.



	Data storage			Data processing			
	Identity (what)	Destination (where)	Time (when)	Path finding	Path choice	Execution	Monitoring
Central computer	X	X	X	X	X	X	X
Conveyor (system)				X	X	X	X
Conveyor unit	X	X	X				

Figure 3.2: Centralization vs. decentralization in MFC

The classification helps to distinguish whether the information about the identity of the conveyor unit, the destination address, and the desired time of arrival at the destination is stored at the conveyor unit itself or is centrally stored. A decentralized storage of this information by the conveyor system would make little sense because the conveyor system is only interacting temporarily with the individual conveyor units and therefore shouldn't save any short-term, conveyor unit-specific data.
On the other hand, the processing of conveyor unit-specific data can very well take place at the conveyor system because a conveyor unit only has little information about the mechanical characteristics of the conveyor system and

would therefore require additional effort be able to find the route through the conveyor system and operate the appropriate actuators by itself.

3.2 Research projects in decentralized material flow controls

Flexible material flow systems are currently the subject of many research projects. The ideas range from increasing the flexibility at the organizational level, to the further technical development of transport components, for example autonomous, driverless transport systems (DTS), all the way to the idea of the "Internet of Things", in which conveyor units autonomously find their way from production to customer - and back again to recycling. (ten Hompel and Nagel 2008).

In the area of the organizational level, the SFB 467 should be mentioned, which is concerned with "versatile company structures for the multi-faceted serial production" (Westkämper, Wiendahl, and Balve 1998). The goal hereby is the development of models, methods and processes to increase the versatility of manufacturing companies. The focus here is on the versatility of the processes and less on the control and techniques of material flow systems.
A large volume of literature exists about segmentation (Wildemann 1988), holonic (van Brussel and Valckenaers 2000), or fractal factories (Warnecke 1995), (Wiendahl 2005).

During the planning and design of production facilities, the demand for an increase in flexibility is answered by the development of modular structures. (Schenk 2002) calls for this throughout the entire intralogistic chain, which ranges from conveyor technology, through storage techniques, all the way to sorting and order-picking systems.
According to (Jünemann and Schmidt 1999) conveying systems are evaluated according to various criteria as, for example, the layout flexibility and the throughput. Comparing these criteria shows a gap for highly flexible conveying means with high throughputs (see Figure 3.3). A part of this gap is to be closed by this dissertation.

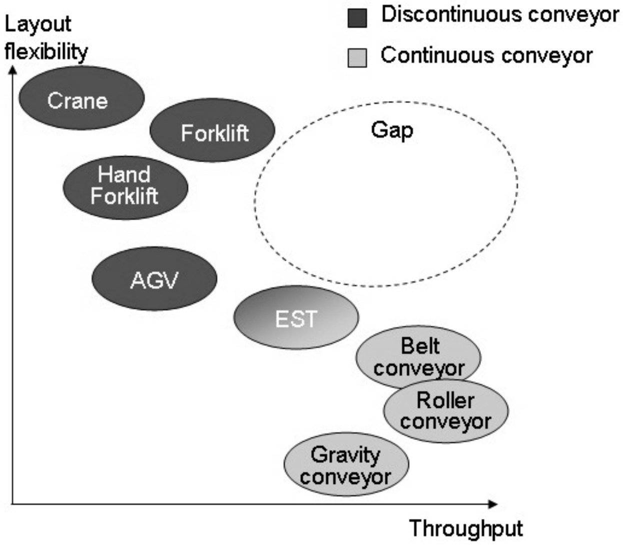


Figure 3.3: Classification of conveying systems for SLBs (< 50 kg)

To develop new concepts for flexible conveyor systems some research centers are engaged in the further development of conventional conveyor technology for a more flexible application. All initiatives are aimed towards a step-by-step decentralization of the control technology and the creation of flexible interfaces. The principle of the *black box* is used to modularize existing technology and to make it versatilely deployable, following the building block principle. Each technical conveyor component is thereby precisely described according to its input and output parameters, as, for example, electrical power supply, control parameters, payload, geometric dimensions and so on, without going into detail about what happens inside the components. This way, different components can be configured together into a complete conveyor system (Wilke 2006).

3.2.1 The Internet of Things

"The procedures for the control of (data) packets within the decentrally structured Internet are known and have proved to be efficient enough to manage the information and communication requirements of the earth's population. This accomplishment qualifies the idea of the "Internet of Things" to become the guide for the revolution of the material flow controls" (ten Hompel and

Nagel 2008).

In the area of information processing, the ”Internet of Things” refers to mostly wireless, self-configuring networks between objects as, for example, home appliances. The first concepts were developed in 1999 at MIT in cooperation with the Auto-ID Center.

In the past few years the best-known initiative for the research of decentralized material flow systems has been advanced at the Fraunhofer Institute for Material Flow and Logistics (IML) in Dortmund under the direction of Prof. Dr. Michael ten Hompel. Under the same name ”Internet of Things”, concepts have been developed on how individual objects of a material flow system (for example conveyor units and conveying means), can achieve full autonomy in seven steps. The goal is to be able to transport goods completely autonomously, without any central control. The basis is provided by equipping all objects with RFID tags, which contain all the information necessary for the accomplishment of the task (Bullinger and ten Hompel 2007), (ten Hompel and Corban 2004).

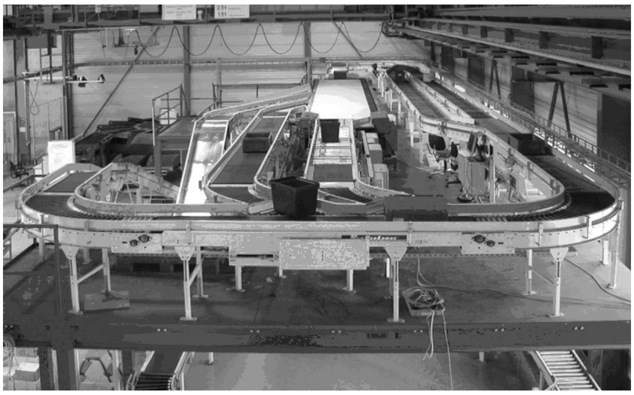


Figure 3.4: Continuous conveyors with decentralized control at the IML for researching the ”Internet of Things”, source: ten Hompel, Libert, and Sondhof 2006

Decentralized control of a piece goods conveyor system

A continuous conveyor system serves as a demonstrator that consists of 36 conveying segments and is controlled with the help of 50 drives and 60 sensors. The system is equipped with 7 IPCs that can simultaneously run a separate control program for each of the 36 segments, each program customized for the respective segment. These were connected via the Ethernet (see Figure 3.5).

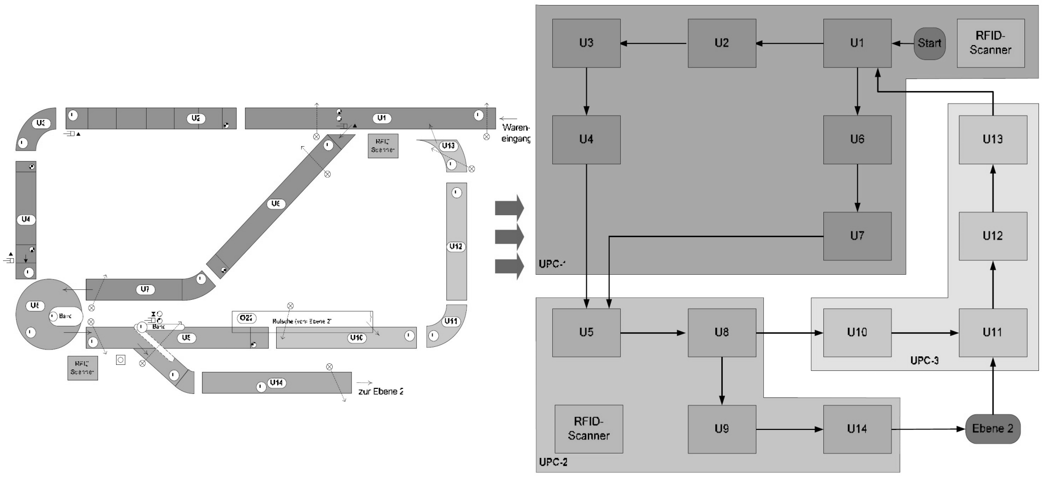
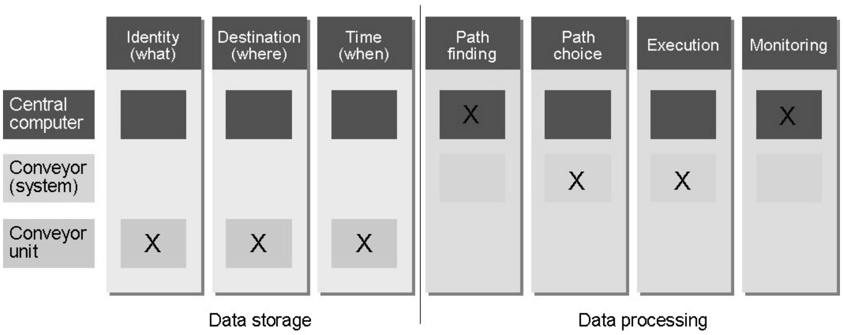


Figure 3.5: Distribution of the continuous conveyor system in 36 segments, source: ten Hompel, Libert, and Sondhof 2006

Four RFID read/write devices with 868 MHz technology were installed and all transport containers were equipped with appropriate tags for identification.

The control program consists of two parts. The logical part was identical for all segments because it was supposed to be executed independently of the hardware. The second part was adapted to the hardware of each segment including the necessary I/O drivers. Because of this splitting, the generation of the individual segment controls was greatly simplified. To make a decentralized decision regarding the path of a container, the control programs have to know the topology of the system. This was manually produced and stored centrally over the network on a workstation PC as an XML file. This is read by the segment controls during start-up and a routing table is locally generated.

The definition of the topology describes routes that are usable in only one direction in the system, which the conveyor units can follow but do not have to. Each route has exactly one starting point and one destination point. A route consists thereby of one or more conveyor sections, each of which corresponds to the successive conveyor segments of the system. If multiple routes are allowed to be used to a certain destination, the decision is made using a priority ranking in the routing table. A differentiation is made between static and dynamic priorities. Static priorities reflect the path length. Dynamic priorities take into account, for example, the current utilization (ten Hompel, Libert, and Sondhof 2006).



	Data storage			Data processing			
	Identity (what)	Destination (where)	Time (when)	Path finding	Path choice	Execution	Monitoring
Central computer				X			X
Conveyor (system)					X	X	
Conveyor unit	X	X	X				

Figure 3.6: Classification of the piece goods conveyor system according to the degree of decentralization

Although the segments possess a high degree of decentralization and assume a large part of the coordination of the conveyor units, a manually-produced and centrally-deposited topology data file still has to be accessed. In addition, the created system can only transport conveyor units on each resulting conveyor section in just one direction. Using the conveyor sections in both directions is not possible. Furthermore, through the use of conventional material conveying technology, there is no advancement in regard to the achievement of a higher mechanical flexibility of the individual segments. Figure 3.6 shows the classification of the system according to the degree of decentralization as per the systematic introduced in chapter 3.1.

Decentralized control based on sensor nodes

The IML in Dortmund is currently also developing a decentralized control of conveyor units based on sensor nodes, where the container already possesses its own intelligence. To be able to realize this, so-called Smart Items were equipped with active RFID components. In contrast to the passive tags in use today, which can only transmit information within the direct operating range of a reading device, these objects are able to exchange information with their environment autonomously.

The concept is based on communication-enabled minicomputers, the so-called sensor nodes, which are attached to conveyor units as well as to active con-node are a small processing unit (microcontroller), a communication unit (radio transceiver), a database, sensors, and a power supply (see Figure 3.7).

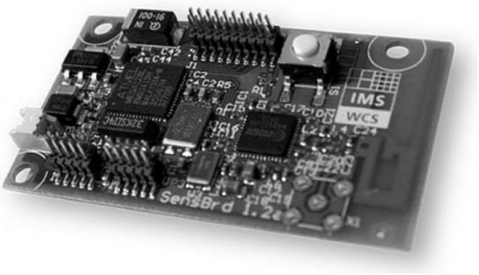
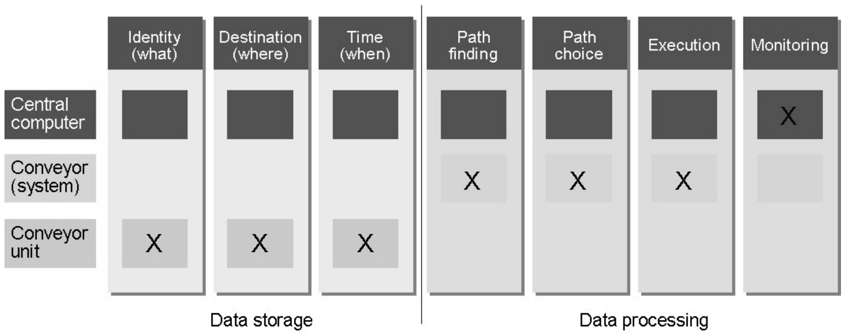


Figure 3.7: Sensor node circuit board

When the system is started up, the conveyor section's sensor nodes begin, through the reciprocal exchange of information, to dynamically build a routing table, on the basis of the information about its local entrance and exit routes, that describes the path network of the conveyor system. This is done using the Link State Routing from IT network technology (see chapter 3.3.4). It must be kept in mind here that each interface is defined as a conveyor input or output and therefore all resulting conveyor sections can always be operated in just one direction.
When a conveyor unit approaches a switch, the sensor node receives a signal from a magnetic switch, which is attached to the conveyor section, that a switch is located a short distance ahead. The sensor node on the conveyor unit then activates its RFID reader and scans the tag that is located on the conveyor section and which contains the ID of the sensor node on the switch and the position. On the basis of the ID, the destination, which is stored in the conveyor unit, can subsequently be transmitted by WLAN to the switch. With the help of the routing table, it makes a decision regarding the transport direction. If the conveyor unit has passed the switch, it is detected by the sensor node on the switch with the help of a photoelectric barrier and a confirmation is sent to the container (ten Hompel and Nagel 2008), (ten Hompel, Schier, and Pöter 2008), (Pöter and Schier 2005).

This concept also uses the conveyor sections in just one direction. In addition, the expense of equipping each conveyor unit and conveyor element with a sensor node with WLAN, RFID scanner, magnetic switch, and processing unit is very high. The generation of the routing tables using WLAN and



	Data storage			Data processing			
	Identity (what)	Destination (where)	Time (when)	Path finding	Path choice	Execution	Monitoring
Central computer							X
Conveyor (system)				X	X	X	
Conveyor unit	X	X	X				

Figure 3.8: Classification of the sensor node-based system according to the degree of decentralization

Link State Routing requires a large amount of computation when there are frequent changes to the layout, and supplementary interfaces are also required in order to physically identify the respective neighbors via a direct path. Besides, the large number of WLAN transceivers can interfere with each other when the volume of data traffic is too high. Figure 3.8 shows the classification of the system by degree of decentralization. It becomes clear that despite the limitation of the one-way traffic and the high costs for the technical realization, the system possesses a higher degree of decentralization than the system introduced previously.

Multi Agent Systems

Another idea for the decentralization of material flow systems goes a step further. Not only does each conveyor segment have its own control software, but also each conveyor unit. The technology used here is labeled by ten Hompel as "agent technology". Each conveyor unit receives its own control software (agents). The necessary information to start the control software is saved in an RFID tag attached to the conveyor unit (Schroer 2005).
An agent possesses the following functions:

- **autonomy**
  Agents operate autonomously, without external manipulation
- **social ability**
  Agents interact with the user and with other agents. The communication occurs on a semantic level (beyond the execution of a set of commands)

- **reactivity**
  Agents are aware of their environment and react on time and in accord with changes
- **pro-activeness**
  Agents not only react to their environment but are also capable of being goal-oriented and proactive

The software agent moves parallel to the material flow. The agent is initialized either on a processing unit attached to the conveyor unit, or on stationary processing units attached to the conveyor route. In the latter case, the agent migrates from processing unit to processing unit according to the material flow. For this a small processing unit is attached to each segment, a so-called node computer. When a conveyor unit enters a segment, the control software on the corresponding node computer is initiated and the agent is born. The agent dies when the conveyor unit leaves the segment. If the agent has made a decision about the route while passing through a segment, he communicates with another program that is running on the node computer, the entity that subsequently controls the respective actuators (ten Hompel 2007), (ten Hompel and Corban 2004b), (ten Hompel, Stuer, and Liekenbrock 2004).

3.2.2 MATVAR

The Institute for Materials Handling, Material Flow, Logistics (FML) in Munich under the direction of Prof. Dr.-Ing. Willibald A. Günther is working on the research project MATVAR (material flow system for variable production segments within the dynamic production environment) which is also concerned with the decentralization of material flow systems (Günthner and Reinhart 2000), (Günthner and Wilke 2002).

Within the framework of MATVAR, a test facility with floor-free, discontinuous conveyors (conventional electric trolley, ETC) was built at the FML, to implement decentralized routing. The transport here is handled by intelligent, individual vehicles following the "taxi principle". A central entity or the conveyor unit itself orders a transport vehicle that picks up the conveyor unit and transports it to its destination. Thereby two different types of order placement can be distinguished.

On the one hand, a central material flow computer can directly assign an order to a transport vehicle. On the other hand, a decentralized order placement strategy can be chosen, which, for example, uses bidding algorithms for the auctioning of available resources.

The vehicle concept can be clarified with the following analogy example: A car driver plans his route with a map and orients himself with the help of street signs. The controller adapts these thoughts and calculates the route locally using landmarks. The planned route is subsequently entered via a hierarchical communication layer into a centrally-filed, manually-produced route map (Waypoint matrix). RFID tags are used as the technical realization of the street signs and are attached next to the conveyor route.
The material flow layout of the ETC was split into active and passive elements. The passive elements are track segments without branches, the active elements can undertake changes of the route (switch). The waypoint matrix can be divided as shown in Figure 3.9 into three levels (Günthner, Chisu, and Kuzmany 2008).

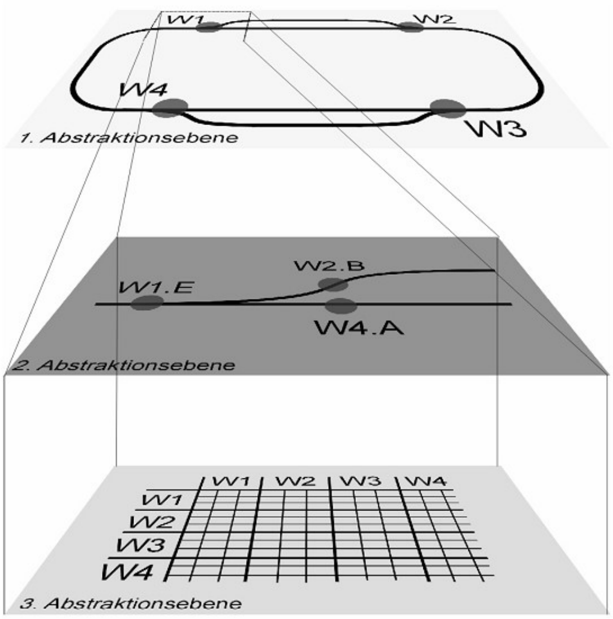
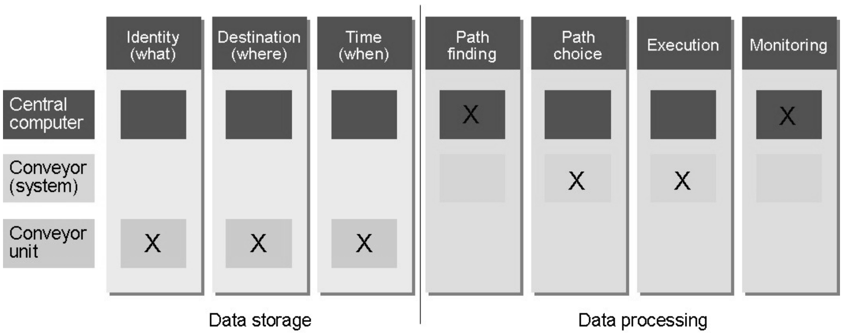


Figure 3.9: Three levels of the waypoint matrix

In the first step of abstraction a global waypoint is assigned to each active module. In the second step of abstraction a local waypoint is assigned to the mechanical interfaces of an active module. In the third step of abstraction the global waypoints are carried over into row and column headers, and the connection including its characteristics are entered into a matrix.
The route planning takes place locally in the vehicle and is carried out using the Waypoint matrix. During the rough planning, the available routes

to the destination via the global waypoints are calculated with the help of graph theory and the corresponding costs are added. Alternative routes are produced by removing individual track sections from the shortest route and then recalculated.
After the rough planning, the vehicle analyzes the generated routes with regard to their internal structure with the local waypoints to finally generate switching instructions for possible changes in direction at the right places along the route. Afterwards the vehicle analyzes the generated routes with regard to their availability for reservation, starting with the shortest route. If this one violates a reservation rule, the vehicle chooses the next shortest route and analyzes it. If a complete route can be reserved, it is entered into the Waypoint matrix and reported to the central computer.
The problem of too many vehicles entering into a certain area and blocking each other was solved in this concept by a complete reservation of the route to the first checkpoint. It is thereby guaranteed that the vehicle can complete its assignment within the calculated time. To avoid a sharp drop in the throughput, additional vehicles are only allowed to also use the reserved route in the same direction (Wilke 2006), (Wilke 2008), (Günthner and Wilke 2003), (Günthner, Chisu, and Kuzmany 2008), (Günthner, Kraul, and Tenerowicz 2008).



	Identity (what)	Destination (where)	Time (when)	Path finding	Path choice	Execution	Monitoring
Central computer				X			X
Conveyor (system)					X	X	
Conveyor unit	X	X	X				
	Data storage			Data processing			

Figure 3.10: Classification of MATVAR by degree of decentralization

In the approach introduced above, the vehicles conduct the route choice and the reservation locally, but the topology information and reservations are stored in a shared, central map. Therefore, as with classic material flow controls, a ”single point of failure” exists. Malfunction of the map computer leads to a malfunction of the whole system (Fay, Jerenz, and Seitz 2008). The sequential access of the individual vehicles to the data base also limits the arbitrary scalability. Furthermore, because of the use of discontinuous conveyors, the maximum throughput is significantly inferior to that of a continuous conveyor system. Figure 3.10 shows the classification of the system by the degree of decentralization.

3.2.3 Transport system in analogy to routing in data networks

At the faculty of automation technology of the Helmut Schmidt University in Hamburg, a further decentralized-control concept was developed where routing mechanisms used in the Internet were adapted to transporting systems. The control concept was developed for an existing luggage conveyor system for airports with identical vehicles (so-called destination coded vehicles, see Figure 3.11), and the performance was tested using simulations. For a technical implementation existing computers on the switches (nodes) can be used. The routing decision is hereby made completely decentralized in the nodes. The passive vehicles are equipped simply with an RFID transponder on which the transport assignment and a unique identification are stored (Fay and Fischer 2004).

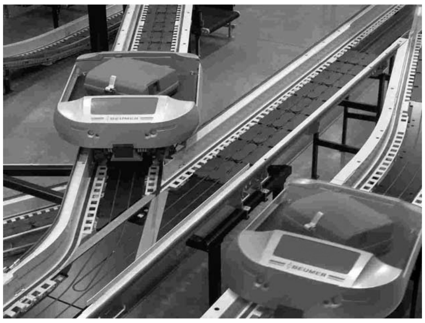


Figure 3.11: Destination coded vehicle (Alstef Automation)

The control concept introduced here assumes a predetermined layout, which is already saved locally in each node in the form of a routing table. When a vehicle reaches a switch (node), the transport destination must first be determined by the node using the transport order saved on the vehicle. Based on the destination, the following node is determined using the routing table stored in the node. When the vehicle has arrived at its destination, the

packet is passed on, a message is sent to the job management and the vehicle is discharged.
If the vehicle has not reached its destination yet, the estimated transport time to destination via the two possible exits of the nodes are derived from the routing table. With the help of a probability function, the so-called Fermi function, both transport durations are converted into a probability from which the next node is stochastically chosen (see Figure 3.12).

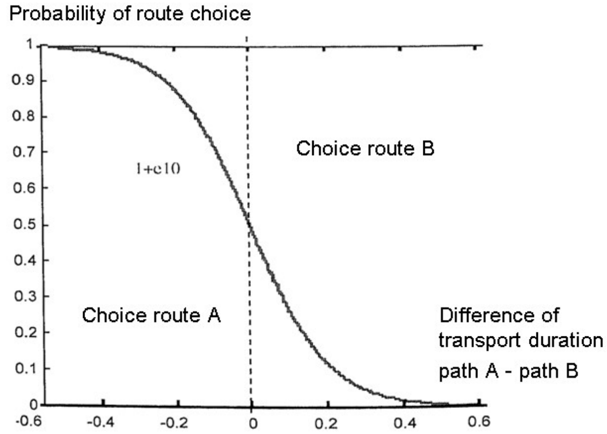


Figure 3.12: Fermi function for the determination of the probability for the route choice

The larger the time advantage (distance) of the faster route to the slower route, the higher the probability that the faster route is chosen. The Fermi function ensures that faster routes are favored, equally fast routes are evenly traveled, and longer routes are occasionally traveled to receive updated information about the actual transport duration on the route.
Various lists are kept in the nodes to keep the necessary transportation times for the route choice updated. When a vehicle travels from node A to node B, its departure time and estimated arrival time is saved in a list in A. When the vehicle arrives at node B it sends a message to node A with its arrival time. The node deletes the vehicle from the list and compares the real transport time with the time for the route saved in the node. If a significant deviation is observed, then the new travel time for this route is communicated to all nodes through a message. There is only a reaction here to delays or early arrivals when a vehicle arrives. If a vehicle is significantly delayed, then the reaction to the changed transport duration can only occur very late. For this reason, as a second mechanism, the lead vehicle is always monitored for delays on

the route. As soon as the vehicle has a delay of a given time and even before it reaches the destination node, the current travel time of the vehicle on this path segment is adopted as the new estimate and communicated to all nodes. With both of these mechanisms the real transport times are always considered and delays are reacted to. If transport times have changed, a recalculation of the shortest route is required. If a node receives a message about a changed transportation time, it recalculates the shortest routes with the help of a graph algorithm taking into account the changed transport time, and updates its routing table.

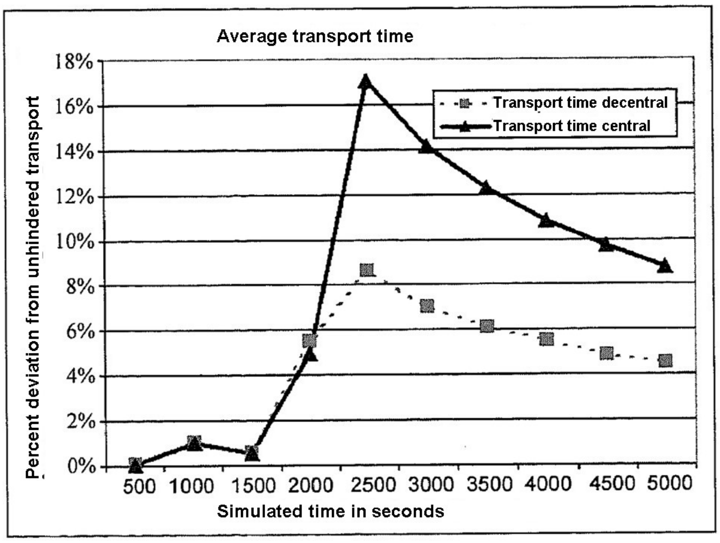
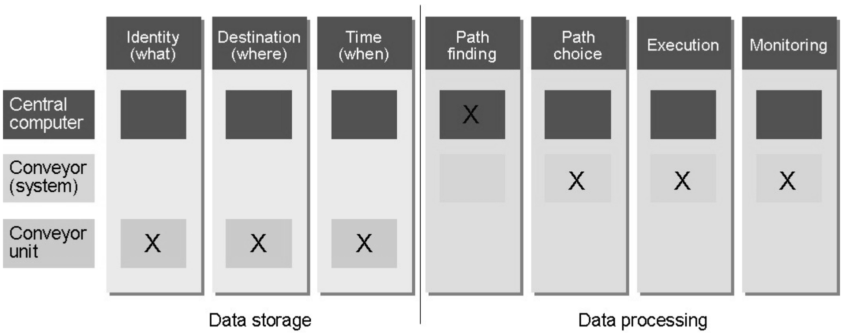


Figure 3.13: Comparison of transporting times of centralized and decentralized controls in case of a hindrance of t=1500 seconds

Running a simulation using a commercial material flow software, it was proved that the decentralized control concept in an undisrupted case delivered at least equally good results as the centralized, but in case of a disturbance it was superior to centralized routing (see Figure 3.13) (Fay and Fischer 2004), (Fay, Jerenz, and Seitz 2008).

Like MATVAR this system requires a centrally stored topology data file, which contributes to a higher possibility of failure. Additionally all routes can only be used in one direction, which again considerably limits the flexibility of the system. In spite of this, the dynamic adaption of the choice of route according to the current utilization is a good approach to improving



	Identity (what)	Destination (where)	Time (when)	Path finding	Path choice	Execution	Monitoring
Central computer				X			
Conveyor (system)					X	X	X
Conveyor unit	X	X	X				

Figure 3.14: Classification of the material flow system by the degree of decentralization

the overall throughput of the system. Figure 3.14 shows the classification of the system according to the degree of decentralization. Unlike the previous systems, this system already analyzes its throughput and adapts itself accordingly. Hence, decentralized monitoring is also performed by this system.

3.2.4 Need for research to achieve complete decentralization

The current developments in decentralized conveyor systems already show a very high degree of decentralization. Nevertheless, a large investment is still required for the installation of the systems because on the one hand the components have to be equipped with much more expensive control hardware (for example, a sensor node costs more than 1000 eur) and on the other hand all of the systems need central components, which as the "single point of failure" can be the cause of major malfunctions. In addition, although much importance is placed on a high degree of flexibility in the control, in all of the systems conventional, mechanical components are used, which were not specially designed for a decentralized application.

To achieve a highest possible degree of decentralization, the concept of the mechanical components has to be fundamentally changed. All components, for example, should be arbitrarily combinable with each other to assure a high level of reusability. With the use of components that are identical in construction, a simple replacement of defective components can be done. Also, the planning and installation of a new layout would be simpler to realize.

For this reason a new concept is introduced in chapter 4 in this dissertation, which allows, through the appropriate design of the controller and hardware, a material flow system to be developed that meets the specified requirements and offers a maximum of flexibility and independence.

3.3 Decentralized control of IT networks

Probably the best-known decentralized systems in daily life are IT networks where data packets are sent locally between individual stations with no central control. Especially the Internet with approximately half a billion networked computers is the pinnacle of decentralized networking since a central controlling unit for so many clients would be unimaginable. The detailed examination of these systems in the following is to serve as a basis for the development of the decentralized material flow system.
The first computer networks were initially centrally organized. The mainframe was connected to multiple workstations. The user was able to access the processing power and the applications on the mainframe while the workstations only handled the input and output devices. The reason for this was the high cost and the large physical dimensions of the computers and the memory. The advantage of the central structure was the possibility to utilize the expensive components to full capacity and therefore take the greatest advantage of the investment.

Because of the rapid development of processing power, however, since the 1980s it has become increasingly cheaper to equip high-performance, personal computers with the appropriate software applications. The advantage was faster access time and the almost unlimited scalability of the network. Today's personal computers are so inexpensive, that mainframes are almost nowhere to be found, except in large computing centers, where extremely large amounts of data have to be stored and processed.
Because of the decentralization of data management and processing it became necessary to connect a large number of individual computers together. Additionally, a centralized coordination of the individual data streams became more and more difficult due to the size of the individual networks. Therefore it was attempted to completely dispense with central components in the network technology of data processing systems (Schoop 1991).

The following chapter explains how the decentralized transmission of information packets in IT systems is handled, in order to establish the basis for how conveyor units could also be transported with decentralized control.

3.3.1 The OSI reference model

The basis of every communication is a defined standard. If two people want to talk to each other they must speak the same language. The rules of every language are defined by grammar and the dictionary. The same is valid for IT networks. The OSI reference model released by the ISO in 1978 forms the basis of all present-day networks.
The aim of OSI is to allow an open communication system that can connect diverse end systems together. It thereby defines the communication behavior in its various levels and functions, and establishes the components of data communication. The OSI reference model divides the communication system into 7 layers where two large groups (the application oriented [5-7] and the transport oriented [1-4] layers) are distinguished (see Figure 3.15)(Elsing 1991). The latter layers contain the functionality of the physical transport of data from transmitter to receiver, which are more interesting as the basis for the control of material flow systems and are introduced in detail below.

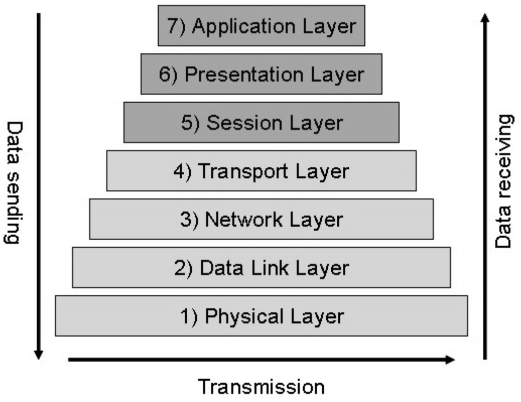


Figure 3.15: OSI reference model

1) *Physical layer*
The task of the physical layer is to transmit digital states over a physical data link, e.g., cable. This is the only level that describes the physical characteristics of a system. Within the physical layer the information is represented

only in the form of single bits. Therefore a protocol is not necessary.

2) *Data link layer*
The OSI model assigns the data link layer two essential tasks: the safeguarding of the individual data bus systems through the use of the appropriate error detection and rectification (collision management), as well as enabling the addressing of various network nodes (destination definition).

3) *Network layer*
The network layer allows the exchange of data between end systems across multiple transit systems. The main task of this layer is the choice of route (routing).

4) *Transport layer*
The transport layer rounds off the tasks of the transport-oriented layers. A protocol in layer 4 converts an end system-to-end system connection into a time-sharing connection. This type of connection is called a logical connection (Harnisch 2007), (Henshall and Shaw 1990).

3.3.2 LAN technology

The LAN (local area network) is the most common IT network system and operates in the OSI layers 1 and 2. All definitions of how the data can be transported over the transmission medium are therein established. The use of a specific transfer method is basically dependent on the network topology. From this certain methods are derived as, for example, the Ethernet.

Joint communication channels
The LAN is a regional network of multiple computers (nodes). The user is generally the operator of the entire network. Every LAN is based on a common medium, usually a cable with which the computers are connected together. The computers alternately use the medium for transmitting data packets.
The fundamental principle of computer networking is based on the locality of reference. This principle says that the communication follows two patterns. When two computers start communicating, the probability that they will communicate with each other again periodically is high. This first pattern is

called temporal locality of reference. The second pattern is called physical locality of reference and assumes that each computer that is connected to the network communicates with nearby computers more frequently than with distant ones. This pattern characterizes the geographic relationship (Corner 2004), (Martin 1989).

LAN topology

The topology of a network describes the structure of the connections of multiple devices to guarantee a joint data exchange. The topology of a network is decisive for its reliability: Only when alternative routes exist between the nodes will the operability be maintained when individual connections are lost. Here IT topologies differ from most material flow topologies because until now these mostly have only one possible path, due to financial reasons. One differentiates between physical and logical topology. The physical topology describes the construction of the network wiring. The logical topology describes the data flow between user devices.
Basically four different basic topologies can be distinguished:

- Star topology
- Ring topology
- Bus topology
- Tree topology

Star topology
In the star topology (see Figure 3.16) all other members are connected to a central member through a point-to-point connection. The central node is

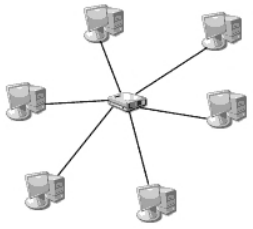


Figure 3.16: Star topology

typically called a hub. When multiple systems are to be connected together, these are also connected through a hub. These central components represent an extremely critical part of the communication infrastructure, and a failure would put the complete system out of operation. Additionally the performance of the system is limited by the performance of the hub. The failure of an end device, however, has no effect on the rest of the network. This structure is easily expandable but requires separate wiring for each member (Martin 1989).

Ring topology
In the ring topology (see Figure 3.17), one member is connected to the next member via a point-to-point connection so that a closed ring is created. The information to be transmitted is passed from member to member until it reaches its destination. Because the communication is deterministic, the predecessors and successors are defined. To avoid interference special addressing techniques are required. Because each member can be used simultaneously as a receiver and repeater, large distances are bridgeable. In the case of a malfunction, the network collapses if the affected member does not have protection switching. In this case a ring redirector is used when a member fails to disconnect this member from the ring and to establish the connection between the two neighbors directly. (Martin 1989).

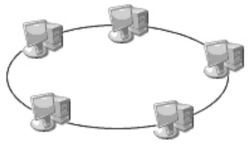


Figure 3.17: Ring topology

Bus topology
The connection of several computers in a bus system consists of a single cable to which each computer is connected (see Figure 3.18). Each member can transmit a signal over the cable that can be received by all other connected computers. So that they don't all transmit at the same time, the computers have to coordinate their sending and receiving activity. Otherwise there would be collisions. This is also a problem with decentralized material flow controls when multiple sources try to send conveyor units over a shared path at the same time. The advantages of bus networks are the low installation

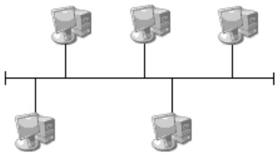


Figure 3.18: Bus topology

effort through the reduction of cabling, and the independence from the functionality of individual members. Even when a node or a station malfunctions, the rest of the system stays intact.
The biggest danger, however, is that a cable break within the main cable can cause a failure of the bus system. Furthermore, the size of the network is limited by the maximum data transfer rate of the bus. The most commonly used network type nowadays - the Ethernet - with its thick and thin Ethernet versions, is counted among the bus topologies (Corner 2004), (Martin 1989).

Tree topology
To allow a more flexible network design, the bus structure evolved into the

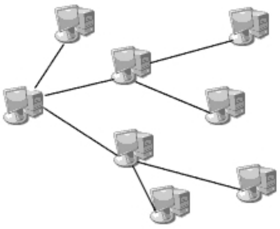


Figure 3.19: Tree topology

tree structure, which consists of networked busses and where multiple nets with the star topology are connected hierarchically with each other (see Figure 3.19). Between two connected end systems there exists exactly one path. A tree topology is achieved by the use of hubs with more than two ports, which separate the individual bus cables from each other. This way, messages can be transmitted simultaneously on different subsystems without any collisions occurring. The tree structure is easily expandable and is therefore especially suitable for complex networks (Martin 1989).

Ethernet

The Ethernet was developed at the Palo Alto research center of the Xerox Corporation at the beginning of the seventies. Today the IEEE (Institute of Electrical and Electronic Engineers) is responsible for the maintenance of the Ethernet standard. An Ethernet LAN conceptually consists of a single cable, the ether, to which multiple computers are connected. An Ethernet is limited to a length of 500 meters and the standard defines a minimum distance of 3 meters between two devices. Because the Ethernet uses a bus topology, a signal is forwarded to all devices. Consequently the transmitting computer uses the entire cable and the other computers have to wait. Upon completion another computer can use the cable.

Because an Ethernet network does not have a central controller that regulates the access to the shared medium, the members take part in a distributed coordination scheme called CSMA. This scheme uses electrical activity in the cable to determine the status. If no computer sends a ”frame” (a complete data packet), the ether does not contain any electrical signals. During the transmission of a frame, the sender transmits electrical signals that are used for the coding of bits. A computer can look for a carrier signal to determine if the cable is currently being used. If not, the computer can transmit a frame on its own. If a carrier is present, then the computer must wait. This carrier scanning process is called Carrier Sense and the concept of using the presence of a signal to determine when transmission is possible, is called Carrier Sense with Multiple Access (CSMA).

When two computers send out frames at the same time, a collision occurs and the two signals overlap each other. Such a collision does not damage the hardware but it damages the transmission so that both frames are received with errors. To recognize such a situation, a transmitter compares the signals in the cable with the signals it generated. If they differ, a collision occurred. If one is detected, the transmitter instantly aborts the transmission, waits for a randomly selected period of time and attempts the transmission again. This is called Collision Detection (CD) and the Ethernet mechanism is identified as CSMA/CD.

If both computers coincidentally choose the same waiting time, a new collision occurs. In this case both computers double the time period from which the random time was taken, and thereby increase the probability of a difference. In this way, with every collision, the probability that the next attempt to transmit will be completed without interference increases by a factor of two.

The algorithm for the doubling of the range is called Binary Exponential Backoff. This basically means that after a collision an Ethernet network can restore itself rapidly because each computer is willing to wait longer between the individual access attempts. Despite an increase of the waiting time with a large volume of data traffic, the network can be scaled as desired without ending in a complete collapse (Corner 2004), (Leon-Garcia and Widjaja 2006).

A distinct difference exists here when the requirements of a material flow system are considered because colliding conveyor units cannot be simply retransmitted. Therefore new algorithms have to be found for a decentralized control. On a transport path in a network, for example, several conveyor units could be transported at the same time over different areas, whereby when two conveyor units are on a collision course, it must be clarified first which one receives the right of way.

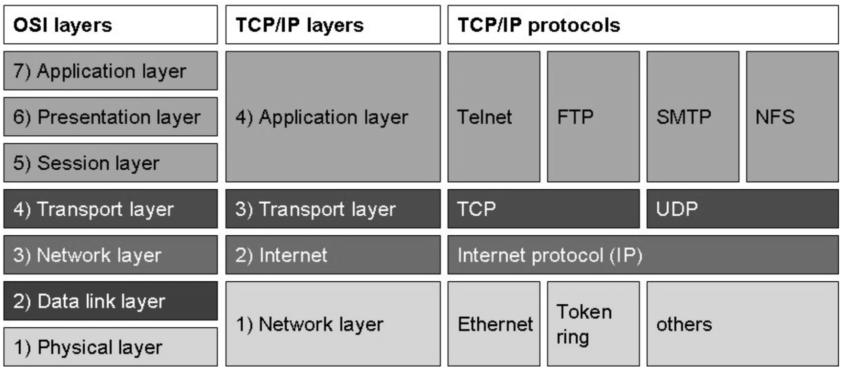
3.3.3 Transport protocol

In the OSI reference model the protocol layers TCP and IP generally lie above the Ethernet, and are also the basis of the Internet. The idea is to define a uniform protocol for the communication between different computer systems. The TCP/IP protocol family is older than the OSI reference model. Hence, TCP/IP cannot be directly mapped onto the seven layers of the reference model (see Figure 3.20). In comparison it covers just four of the protocol layers; however, the individual protocols have a larger functional range (Schürmann 2004), (Leon-Garcia and Widjaja 2006).

TCP/IP is subdivided into:

- TCP (*Transmission Control Protocol*): transport layer
- IP (*Internet Protocol*): network layer

Transmission Control Protocol
TCP is a protocol for a secured connection between two systems and defines the method by which data is transferred between computers. A proper transmission of the data is ensured by the verification of received data packets. If the sender does not receive verification within a specified time, an error is assumed and the same data packet is transmitted again. Also, TCP guarantees the preservation of the sequence order between the individual data packets of a data transmission.



OSI layers	TCP/IP layers	TCP/IP protocols			
7) Application layer	4) Application layer	Telnet	FTP	SMTP	NFS
6) Presentation layer					
5) Session layer					
4) Transport layer	3) Transport layer	TCP		UDP	
3) Network layer	2) Internet	Internet protocol (IP)			
2) Data link layer	1) Network layer	Ethernet	Token ring	others	
1) Physical layer					

Figure 3.20: TCP/IP protocol family

Each TCP connection is clearly defined by two end points. An end point is represented by an ordered pair consisting of IP address and port. With the help of the IP addresses the computers participating in the connection are identified, and with the help of the ports the programs that are communication with each other are identified.

When establishing a TCP connection the three-way handshake is used. The computer that wants to establish the connection (client) sends a SYN (synchronize) packet to the receiver with a sequence number x. The sequence numbers are important for ensuring a complete transmission in the right order without duplications. The starting sequence number is a random number whose generation is dependent on the particular TCP implementation. It should be as random as possible to avoid security risks.

The receiving station (server) receives the packet. If the port is closed, it answers with a TCP-RST to signalize that a connection can not be established. If the port is open, it sends in return its own SYN packet containing its starting sequence number y, which is also random and independent of the starting sequence number of the client. At the same time it confirms the receipt of the first SYN packet by increasing the sequence number x by one and sends it back in the ACK (acknowledgement) part (see Figure 3.21).

The client confirms in a final step the receipt of the SYN/ACK packet by sending its own ACK packet with the sequence number $y+1$. This procedure is also called a ”Forward Acknowledgement”. Moreover, the client sends back the value $x + 1$ for security reasons. This ACK packet is received by the server. At this point the connection is established (Leon-Garcia and Widjaja 2006).

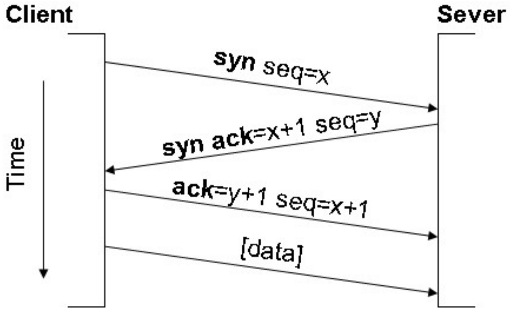


Figure 3.21: Establishing a connection with TCP

During data transmission the server sends a receipt for every packet received. The client keeps a copy of the packet and awaits the ACK before sending out the next packet. TCP starts a timer before each data transmission. If the timer expires before the confirmation is received, the sender transmits the data again because the time-out is interpreted as data loss.
Using the sliding windows technique, several packets can be transmitted until the first ACK arrives. Therefore, a single ACK at the end of the transmission of multiple packets suffices. TCP divides the data stream into segments, which are chosen without regard to the size that is used by the application program. The key benefit is the efficiency as opposed to the transmission of single packets one after the other (Harnisch 2007).

Internet Protocol
IP takes over the data packets from the higher level and transmits them over the network. The main task of IP is the association of data and nodes and makes the routing possible, that is, the search for the ideal route between the two stations involved. IP embodies the idea of an unreliable delivery of packets and the associated idea of routing. The security of the delivery has to be assumed by other levels.
A packet could get lost, duplicated or transported beyond the boundaries, but the Internet will not register this event and will notify neither the sender nor the receiver. Each packet is treated independently from the others. The Internet undertakes ”serious” attempts to deliver, meaning it will not ”capriciously” discard any packets (Harnisch 2007), (Leon-Garcia and Widjaja 2006).
In order to create a large network, the Internet uses a standardized addressing scheme. Each participant is assigned a protocol address. This address is used by users, application programs, and most protocols for communication.

Within the TCP/IP family, the Internet Protocol defines the addressing. The IP standard divides each Internet address into two parts: the prefix of an address is the identifier of the network to which the computer is connected. The suffix represents the identifier of the computer concerned. To guarantee the unique assignment of addresses in the Internet, the identifiers are assigned by a central authority. After the assignment of a prefix, the local network manager can assign a distinct suffix to each connected host. A long prefix allows for many networks but it limits the size of the individual networks. A long suffix means that many computers can be connected to each physical network but the total number of networks is lower. (see also NAT - Network Address Translation) (Corner 2004).

Data packet switching
Instead of using leased lines to connect two computers directly to each other, a network often includes switches to which the individual computers are connected. A switch detects the reception of data from a computer and sends it on at high speed through other switches to the receiving computer. A switch is conceptually a small computer (with a processor and memory) that is exclusively used for the sending and receiving of packets. That way it can use various transmitting media as, for example, copper wire or optical fibers. Furthermore it is able to buffer several incoming packets (Store-and-Forward Switching) in case the lines are busy at the moment.
A switch also independently determines its position in the network and produces routing tables to find the fastest routes to other nodes. The transmission of packets over partial sections depends thereby neither on the source of the packet nor on the individual stages up to the relevant switch, but only on the destination of the packet. This concept, called source independence, is one of the most important fundamentals of networks and is also used in this dissertation for the development of the decentralized material flow control.
Because of the source independence the routing mechanism of a computer network can operate compactly and efficiently. Since no source information is required for the routing, only the destination address must be extracted from the packet. (Corner 2004), (Martin 1989), (Leon-Garcia and Widjaja 2006).

3.3.4 Routing in networks

Routing algorithms can be categorized in three dimensions:

- central - decentral (or local)

- static - dynamic
- nonadaptive - adaptive

With **central** routing algorithms, the complete topology of the network is to known a central control unit. Correspondingly, with **decentralized** algorithms only parts of the network are known to individual control units (e.g. routers).

Every entry in the routing table with **static** algorithms is done by hand. A malfunction of individual routers does not lead to a revision of the table. **Dynamic** algorithms develop the routing tables completely on their own and adjust these when a router fails. In the Internet, dynamically-created tables are frequently manually adapted to exclude certain network segments.

Adaptivity refers to the characteristic of the algorithm, to adjust the routing table to reflect the load on the network. Therefore not only changes of topology are considered but also the current traffic.

Routing is basically a problem of graph theory. The labeled nodes of the graph can be hosts, switches or routers. The lines of the graph correspond to the network connections. Each line corresponds to expenses that indicate whether or not it is desirable to send traffic over this route. To be covered below is the most important graph algorithm, the *Dijkstra-algorithm*, which serves as the basis for the routing algorithms (Leon-Garcia and Widjaja 2006).

Dijkstra algorithm
The Dijkstra algorithm is one of the most well-known graph algorithms for finding the shortest paths and was published by Dijkstra in 1959. It is based on an iterative expansion of a set of shortest paths to all other nodes and finds an optimal solution for nonnegative weights. Figure 3.22 shows a simple weighted graph. The line (S, U) has the weight 10. If the weights are perceived as cumulative costs, then it is cheaper to go indirectly via X from S to U (total cost = 8) than to use the direct line.

Per node, a value D is saved which is given the value 0 for the start node and at the end of the procedure should save the correct distance value to the start node. During computation this variable contains intermediate values and is set to the value ”infinite” at the beginning. Figure 3.23 (1) shows the result

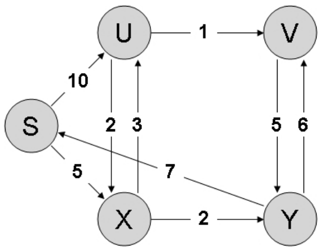


Figure 3.22: Weighted graph

of the initiation phase for the start node S.

The priority queue Q has the following form after the initialization:

$$Q = \langle (S:0), (X:5), (U:10), (V:\infty), (Y:\infty) \rangle$$

After the first cycle the algorithm calculates the state shown in 3.23 (2). Each node that is directly accessible from S is tagged with the weight that is assigned to the direct line, and S itself is removed from Q.

$$Q = \langle (X:5), (U:10), (V:\infty), (Y:\infty) \rangle$$

The second cycle processes the costs with the minimum distance value in the priority queue, in this case node X with the value 5. This value is the actual final distance value from X because every other path to X would have to go over (S, U) and would lead to a distance of at least 10. Figure 3.23 (3) shows the result and Q is adjusted as follows:

$$Q = \langle (Y:7), (U:8), (V:\infty) \rangle$$

The adjustment of the value of $D[U]$ is hereby observed. It is noted that the indirect path via X requires lower costs than the direct line (S, U).

In the third step node Y is processed. Figure 3.23 (4) shows the result.

$$Q = \langle (U:8), (V:13) \rangle$$

In the fourth step node U is processed, see Figure 3.23 (5). Q now contains only one node:

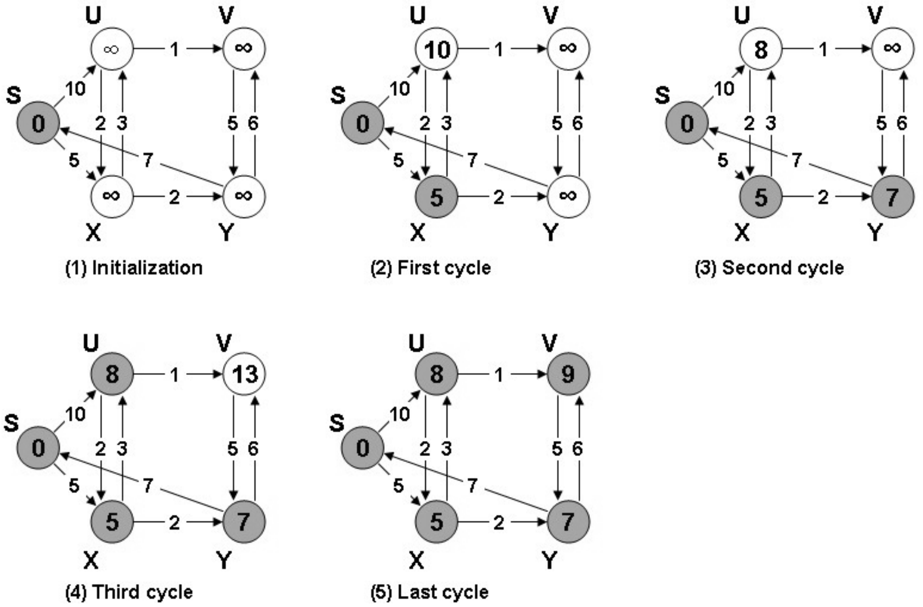


Figure 3.23: Functioning of the of Dijkstra algorithm

$$Q = \langle (V : 9) \rangle$$

The processing of the last node does not change any of the distance values. Therefore the end result is reached (Saake and Sattler 2006), (Furmans and Arnold 2006).

The Dijkstra algorithm only works when no negative weights are to be found. Otherwise a local unfavorable, following line would subsequently become a better connection through a negatively weighted line that comes after it. In this case the Bellman-Ford algorithm is better suited but due to its lack of relevance in network controlling, it is not described any further in this dissertation (Saake and Sattler 2006).

When applying these two algorithms to network technology, adjustments have to be made. Since the velocity of the information through the ether is close to the speed of light and the processing time in each node is on the average the same, the lines are set, for example, to *cost* = 1. For the final generation of the routing table, two different procedures based on the two algorithms are also differentiated: *Distance Vector Routing* and *Link State Routing.*

Distance Vector Routing

Distance Vector Routing (DVR) is a dynamic routing algorithm that functions according to the principle "Let your neighbor know how you see the world". It is used by routers in data packet transmitting networks and is primarily implemented in the internet as the Routing Information Protocol (RIP) or Interior Gateway Routing Protocol (IGRP).

In DVR, each node creates a one-dimensional array that contains the respective "distance" in the form of costs to all remaining nodes, and communicates this vector to its direct neighbors. At the beginning each node knows just itself and its direct neighbors. A failed line is assigned infinite costs. Receiving nodes each add the additional distance from the sender to themselves to the entries. Afterwards, they update their own array with the new information, if new or better paths were found. Is this the case the new array is also forwarded to their neighbors.

The disadvantage of the Distance Vector Algorithm is the *count-to-infinity problem.* When circular paths exist in the network, then the nodes potentially know several exits to other nodes. If a node fails, then the direct neighbors set the distance to infinite and report this to the neighbors farther away. If these know other exits to the failed node, then these routes are not deleted, but rather reported to the direct neighbor as a new alternative. This node in turn adds the additional route and informs the rest of the neighbors. Each time the message has gone around the circle, the distance of the circle is increased. To stop this incrementation, the RIP protocol limits the maximum number of hops (forwards) to 15 (Hedrick 1988), (Rekhter, Li, and Hares 2006).

Because the Distance Vector procedure is used in this dissertation as the basis for the independent generation of the topology information within the material flow system, the precise message transport procedure is discussed later in my own approach. The further development of the Distance Vector method basically lies in the fact that with DVR, the array is completely transmitted to the neighbors after a defined amount of time, whereas with the control developed in this dissertation the individual routing entries are sent out consecutively in short intervals. Furthermore, the identities of the nodes that have already been processed are included when the respective routing entries are sent, so that the *count-to-infinity problem* can not arise. Both enhancements make the algorithm capable of processing an infinite number of nodes with a large number of circular paths. The somewhat longer convergence time of a few seconds is acceptable because the physical construction of the material flow systems or later modifications require much more time.

Name	Distance Vector protocol	Path vector protocol	Link State protocol
Example	RIP	BGP	OSPF, IS-IS
Each nodes stores...	...for every port the distance to every destination	...for every port the distance to every destination and the shortest path	...all existing connections incl. their lengths, and for all destinations the optimal route
Each node knows...	...only its neighbors	...only its neighbors	...the complete network topology
Storage req'd	very low	very low	high
Computing time	low	low	high
Remarks	Due to "count-to-infinity" not used in the Internet any more	Avoids "count-to-infinity" problem by storing the paths to the destination; Standard in Internet	Used only in limited networks due to high memory and computing requirements

Table 3.1: Comparison of data routing mechanisms

Link State Routing

With Link State Routing it is assumed that each node can detect the status of the connection to its neighbors and therefore the costs. The basic principle is the following: Let the world know who your direct neighbors are. When it is certain that all of this information has been propagated to every node, then all nodes possess adequate knowledge of the network to create a complete mapping with the help of the Dijkstra algorithm. Consequently, Link State Routings are based on two mechanisms: dependable spreading of information and computation of routes out of the sum of all this information.

Dependable **broadcasting** is thereby a process that ensures that all participating nodes in the routing protocol receive a copy of the information from all other nodes. In the process a node transmits the information over all its directly-connected routes and every node that receives this information forwards it accordingly. This process is continued until all of the information has reached all nodes.

The Link State Packets (LSP) contain the ID of the node, a list of the directly-connected neighbors, a sequence number, and the lifespan of the packet, which

ensures that the information is up-to-date. Upon receiving an LSP, the node verifies the sequence number with an existing LSP from the transmitting node, if available. If the sequence number is higher, it means that the LSP is more current and must be processed.
As with RIP each node produces LSPs under two conditions: either when a periodic timer expires or because of a change in the topology. This is the case if a direct neighbor has failed or has been newly added. The most common realization of the Link State Protocol in network technology is called Open Shortest Path First (OSPF) (Moy 1998), (Aurand 2000).

Dependably designing the broadcasting has so far proven to be quite difficult because the data traffic in large networks should not be underestimated. Additionally, the calculation of the routes using Dijkstra after the arrival of all LSPs in large networks involves a potentially time-consuming effort. Ultimately, the application of Dijkstra calculates only the shortest routes, whereby transports over longer routes could definitely be of interest when the shortest route is blocked by opposing traffic (Peterson and Davie 2007).

3.3.5 Decentrally controlled information vs. material flows

To be able to use the concepts of message transmission within IT networks as a basis for a decentralized control of conveyor units in material flow systems, it is necessary to point out the differences of both types and to derive the consequences for the processing. These differences are listed below:

Packet splitting
When transmitting data, the information is partially split into multiple data packets and transmitted consecutively. The purpose is to be able to make the line available at any time for other, possibly more important, packets. Such a splitting is not possible with the transport of conveyor units.

Speed
"Throughputs" of network structures are measured in bit/s. Depending on the transmission medium these can be 4800 bit/s up to 480 Mbit/s and more. Electrical and optical signals basically move almost with the speed of light through the ether (galvanically bonded cables) and therefore the real speed of a data packet and the distance to be covered is not the limiting factor as compared to the processing or the coordination time within the nodes. Dur-

ing the transport of conveyor units, the speeds are limited to a few $\frac{m}{s}$ and therefore the main concern is the optimization of the path length. The time required for acceleration and deceleration also comes into play. Because the communication between single nodes is exponentially faster than the actual transport of the conveyor units, it is assured that with simple algorithms enough time remains to prevent collisions through communication and information processing while the conveyor units are still in motion.

Transmission distance
Electric signals weaken over large distances and have to be regularly renewed or amplified. This effect does not arise with conveyor units.

Collision
By far the most important difference lies in the absolutely necessary prevention of all collisions during the transportation of conveyor units. It is not only a question of preventing the collision of two packages approaching each other on the same path from opposite directions, but also of preventing deadlock situations (see chapter 4.2.5) in which several conveyor units impede the continuation of each other's transport. This constraint does not apply to data transfer because each router holds the data packets back, to resend them again as necessary. Hence the possibility exists to copy packets as often as required and to send them out in multiple directions at the same time. Paying attention to this difference is especially important in the development of decentralized control systems.

Path allocation
Another distinction is the allocation of route sections. While an electrical signal immediately occupies all galvanically connected route sections, a conveyor unit exists only in its current position. Therefore it is possible, at least in the same transport direction, to transport several conveyor units one behind the other at the same time.

Buffering in nodes
Current routers and switches are capable of buffering several incoming data packets until the next line section is free. Nodes in material flow systems usually don't have this ability. Therefore it must be ensured before dispatching that a deadlock will not occur.

Optimization objective

While the main concern during the transport of data lies in optimizing the efficiency of the ether, the optimization objective during the transport of conveyor units lies in the optimization of the path and in minimizing the travel time.

Network size

While physical material flow networks normally do not have more than 100 nodes, approximately 500 million computers were linked together in the Internet at the end of 2007 according to the "Internet System Consortium".

Tabelle 3.2 summarizes the principal differences between the transport of data packets within an IT network and the transport of conveyor units in a material flow system.

Attribute	data transmission	material flow
Packet splitting	possible	not possible
Speed	approx. 300.000.000$\frac{m}{s}$	approx. 1$\frac{m}{s}$
Consequence of a collision	low: resend the data	high: system is blocked
Simultaneous path occupation	not possible	possible in the same transport direction
Buffering in nodes	possible	not possible
Focus of optimization	utilization of the ether	minimization of path distance
Network size	internet approx. half a billion	usually less than 100 nodes

Table 3.2: Comparison of electronic data transmission and material flow

4 Completely decentralized autonomic continuous conveyor system

4.1 Overview and general assumptions

The objective of this dissertation is to set foundations for highly flexible material flow systems, which are distinguished by a completely decentralized control system and a high degree of autonomy. Initially, the basic requirements must be determined.

The basic requirements then determine the specifications of the control system and the ports at the respective system boundaries. After determining these specifications, the control algorithm will be described in detail, which processes all incoming signals from sensors and communication ports as well as outgoing signals to actuators or communication ports.

4.1.1 Requirements for a completely decentralized system

Complete decentralization

This dissertation distinguishes itself from other existing development projects of decentralized control systems for material flow systems, because a system was developed, whose conveyor system (CS) no longer requires any central infrastructure. The system consists of several subsystems, hereafter called "modules", which each undertake a part of the conveyor process and can forward the conveyor unit (CU) within their physical boundaries from a defined entry point (entry port) to a defined exit point (exit port). The overall layout is built by connecting the modules together. The modules generate all required information by communicating with each other, in order to forward the CU to its correct destination (see Figure 4.1). Hence, every module has its own unique address, comparable to the MAC addresses of IT network users.

A superordinate level must exist, which defines the injection point of the CU into the system (source) as well as its point of exit (sink). As this is not a subject of this dissertation, a basic requirement is established, that a CU with its respective identification (CU ID) can be injected into the system at any point and that it carries the address (Dest ID) of its final destination (sink module). This information is recorded by an identification system, e.g., a barcode or a RFID tag. The destination address is defined by the sink description. This description can either be the name of the destination module (module ID) or any other sink description that can be associated with a module ID. (In the case of an airport luggage conveyor system, for example, the module ”21” could designate the departure gate with destination Madrid, LH756). This designation, though, has to be available to the modules. A more detailed examination of the interfaces between the system and the outside world is undertaken in chapter 4.5.

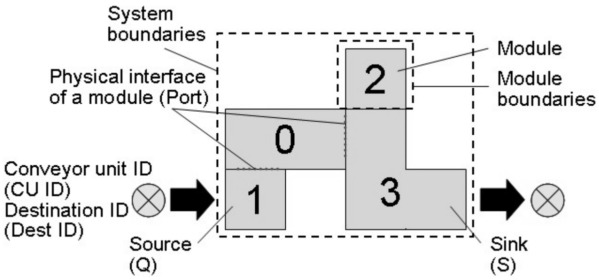


Figure 4.1: Schematic diagram of the system

Autonomy

An autonomous system not only has to accomplish its main transportation task but also react to external factors and changes. On the one hand, these could be disruptions resulting from incorrect operation or wear and tear. On the other hand, the system has to be able to react to planned changes, e.g., a capacity increase or a change of layout. The flexibility or changeableness of material flow systems has already been researched in detail by Prof. Dr.-Ing. W.A. Günthner. Hereby, flexibility is subdivided into flexibility of layout, flexibility of capacity and flexibility of transported material (Günthner, Heinecker, and Wilke 2002). While the latter two are defined mainly by the mechanical structure of the system and therefore are not a subject of this dissertation, the flexibility of layout, however, is mainly defined by the functionality of the control.

Hence, the system presented here needs to be able to adapt itself automatically to any change in the layout. This also applies if one module fails and therefore a possible route is suddenly no longer available.
The biggest challenge of the system though is to guarantee, despite full decentralization, that CUs arrive safely and quickly at their destination. The modules do not have an overall view of the system, which complicates the coordination and increases the danger of two CUs colliding or blocking each other's way.

Basic requirements
The basic requirements of the developed system are established as follows:

- The system consists of many single conveyor means (modules), which due to their arrangement form the CS
- Each module has its own unique address (see MAC address in IT) by which they can be distinguished
- The modules have defined interfaces (port) through which they are connected to their neighbors for communication and also for the physical handover of CUs
- The modules need to react automatically to any layout changes
- The destination address is tagged to the CU (Dest ID) and accompanies the CU
- Every conveyor section can be run in several directions

4.1.2 Determination of the physical features

Further features of the module and CU geometry are established to help with the development of the control system. Hence, hereafter the modules will be shown as square elements with four ports. The developed control system, though, is applicable to any number of ports. Modules are therefore always combinable with 2 to n ports.
A port is classified as "passive" if it is not adjacent to another module. It is very important to consider that all technical requirements within the module have to be complied with, i.e., that an arriving CU can be transported from one port to any other port to be transferred to the neighbor modules.

Ports can be an entry port as well as an exit port, depending on the route of the CU. Furthermore, each port has a communication channel to communicate with adjacent modules. CUs may only be injected into the system via a module that is not simultaneously a sink module for another CU, as otherwise collisions or blockages can not be avoided. But it is possible to use modules as source and sink at different times (see Figure 4.2).

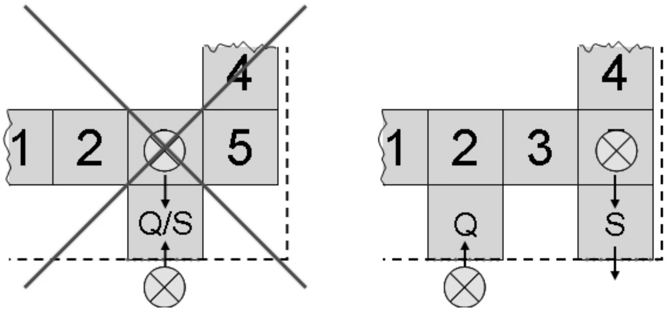


Figure 4.2: No simultaneous use as source and sink

Furthermore, each module must be able to recognize an arriving CU at any time. It is assumed that the modules are equipped with appropriate sensors (see chapter 5).
An identification system has to be implemented in order to assure a continuous locomotion of the CU. This system needs to be able to read the sink description (Dest ID) and the CU ID of the CU prior to approaching the point of decision for a possible change of direction. Should this point be passed before the route has been decided, the CU has to be reversed, which can result in a slow down or stall of the material flow. It is established that a module can only process one CU at a time within its module boundaries. This is necessary because a module in its function as subsystem is not designed either mechanically orwith appropriate sensors for the coordination of several CUs within its subsystem boundaries. This way the maximum capacity is reduced because the minimum distance between two CUs is at least the length of one module (see Figure 4.3; $s - s_0$).

For an adequate scale when employing many modules of the same type, each module should have the same control procedure. The same programming of each module collectively secures the system performance as a whole. A change of the control logic is only necessary if different modules are employed, which have to administer a different number of actuators or sensors for the transport

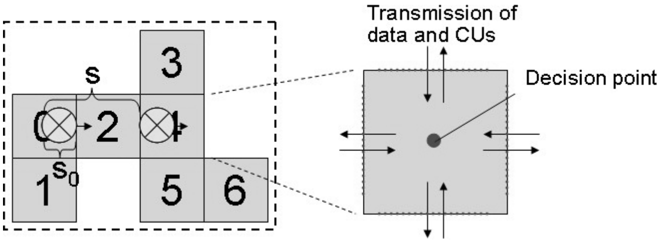


Figure 4.3: Left: module arrangement, right: module structure

of CUs. But in cases like this, the basic control logic will still be maintained. Furthermore, it is established that those modules that form the boundary of the system, i.e., that function as a source or sink, may only have one neighbor module. This is necessary to prevent a CU from being injected into the conveyor section at any point. Were this is the case, the CU could block the CS because other CUs could already be travelling in the opposite direction.

In summary, the following additional requirements for the developed system are established: (see Figure 4.3)

- A module can have 2 to n ports. For simplification, only square modules with four ports are used in this dissertation
- Every module is equipped with the same control algorithm
- Each port has a communication channel linked to the adjacent neighbor module. If no module is adjacent, this port is ”passive”
- A module cannot be source and sink at the same time
- Within the module boundaries only one CU can be processed at any one time
- Source and sink modules are situated at the system boundaries and have only one module connecting to the system

4.1.3 Application example

An application of a control system following the above described requirements is explained here with a brief example.
A plant operator could join several modules to forman adequate CS layout according to the position of the sources and sinks within the production line. After activation, the modules will start automatically and without further setting up to search for their neighbors and their own position within the

CS and autonomously generate a routing table with all possible routes from source to sink modules.

The CUs will be tagged with their unique CU ID and a sink description, which can be allocated to the address of a module (module ID). As soon as a CU is injected into the CS, the modules start to transport the CU in the direction of the sink module. Once arrived at the final destination, the CU will be released into the outside environment. This could be by transferring the CU to a different CS or by manually removing the CU from the system. In case of a change in the layout, the system reacts automatically and searches for alternative routes. Due to the coordination of the modules, the transport of several CUs at the same time is possible.

4.2 Control concept

Being a fully decentralized system, the characteristics of the system as a whole are determined by the characteristics of each module. The modules have to react to all situations in a way that the system always remains functional.

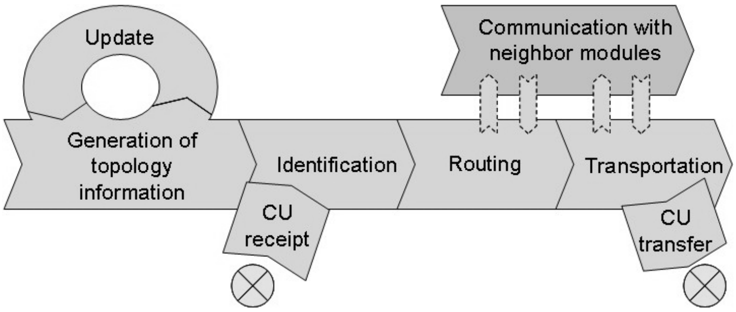

Update
Communication with neighbor modules
Generation of topology information
Identification
Routing
Transportation
CU receipt
CU transfer

Figure 4.4: Process overview of the control logic of a module

The control logic of the modules can be divided in four process steps (see Figure 4.4):

In the first step, the module must gather information about the **topology** of the CS. Initially, when the system is first constructed, the modules do not yet know their position in the system or their neighbors. The module saves the generated topological information in a routing table, which is updated on a regular basis, in order to react to changes in the system. Such changes could be the failure of another module or the expansion of the system by adding more modules.

On arrival of the CU at the port, the module recognizes the arrival and transports the CU in the direction of the decision point in the middle of the module. At the same time, the CU ID and Dest ID are being read. On positive **identification**, the routing is started. Should this process take longer, the CU is stopped at the decision point of the module.

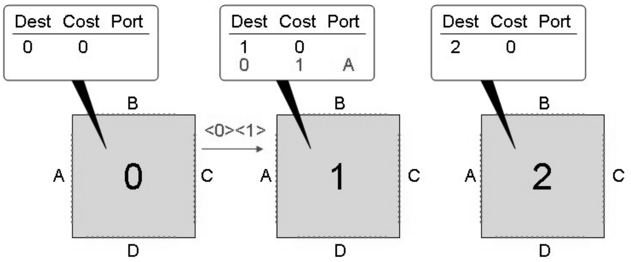
During the **routing** process, the exit port of the shortest available route for the further transport is calculated. This is the biggest challenge for the module as not only the shortest route has to be calculated but also other CUs, that are already in the system have to be considered. The shortest route, for example, could already be transporting a CU in the opposite direction. In such a case, the CU either has to wait or a new alternate, possibly longer route has to be calculated. The decision has a big impact on the potential throughput of the system and strongly depends on the layout.

After a successful routing, the CU is transported to the next module via the chosen port. The next module, however, must be ready to accept the CU. Furthermore, the transmitting module needs to be ready again after the **transmission** to accept the next CU. Prior to the transport, it must be checked if the forwarding of the CU could cause a deadlock. This is especially necessary in systems with a high filling rate or circular paths.

4.2.1 Decentralized generation of topological information

The generation of topological information comprises the gathering of information about the physical location of the modules in relation to other modules in the form of a graph. This information is necessary for the routing of the CUs. Because the modules are connected via their ports, it is logical to also use this connection as an information interface. This way, a module can allocate received information directly to a conveyor direction. The application of a bus system is not sensible because the relative location of the sender is not automatically included. On the other hand, a serial connection to adjacent ports is suggested. When the layout is physically changed, the modules are separated and re-connected to different ports. (This technical difference is a crucial distinction from other developments of decentralized control systems, which are mainly based on bus systems, see chapter 3.2).

Initial generation of the routing table
In chapter 3.3, two completely different procedures (Distance Vector Algorithm and Link State Algorithm) were introduced for the generation of routing tables based on network technology. The basis for compiling routing tables in the new control system is the Distance Vector Algorithm (DVA) with its extension to the Path Vector Algorithm, because it requires significantly less calculation in the modules. Furthermore, the Link State Algorithm requires a broadcast of the information, which is very complex due to the existing serial connections, because information can only be transmitted from one neighbor to the next.
Prior to its activation, a module has only one fixed entry about itself in the routing table. This entry consists of its own identity Immediately after its activation, the module starts to transmit this information to its neighbors in random intervals (e.g., 0-3 seconds). The information consists of the module ID and the cost information, which is the length of the route between the start module and the recipient. As this first entry is being transmitted by the start module, the costs to the first neighbor have the value "1".
Hereafter, all messages transmitted between the modules are called *"tokens"*. There are tokens to gather topological information (topology token), to reserve routes (reservation token), to avoid blockages (deadlock token), to report the failure of a module (reset token), and to finally release the CU (clearance token).
When the neighbor receives the topology token, it amends its own routing table with the new information and the port through which the topology token was received (see Figure 4.5). The port is also set to "active". This information is necessary to later recognize the removal or failure of a module.



Dest	Cost	Port
0	0	

Dest	Cost	Port
1	0	
0	1	A

Dest	Cost	Port
2	0	

Figure 4.5: Transmission of the module ID of module 0

In order to inform other modules which are further away about the location of the sender, the receiving module sends the topology token to all other active ports after having amended its routing table (see Figure 4.6). The costs are thereby increased by "1". The next receiving module will now know in which direction and how far away the source module is. In this way the information about the location of the originating module is disseminated throughout the entire conveyor system.

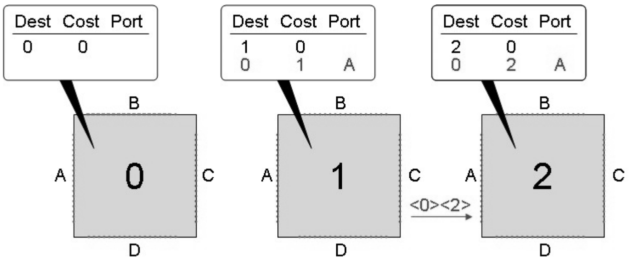


Figure 4.6: Forwarding of the module ID of module 0

If the modules are connected in a circle, there is the danger that the topology token is transmitted infinitely from module to module and the routing table receives an infinite number of entries for the same destination. This can be prevented by only processing a received topology token if no entry with the same or lower costs for the same port has been previously received. This way, a topology token will be immediately discontinued once it starts to go in circles (see Figure 4.7). Simultaneously, entries with higher costs will be overwritten if a topology token has found a shorter route. The routing tables improve themselves step by step until no better entry is received for a port. Now the routing table is complete and the system is ready to fulfill transportation tasks.

Figure 4.7 shows that module 1 receives a second entry, although the CU would travel in circles if it chooses this route (see route entry: 0-5-B of module 1). In order to avoid such invalid route entries, all modules that have already been processed, are also recorded in the topology token. If a token arrives at a module which it has previously passed through, it will be stopped (see Figure 4.8). This logic is also applied to avoid the "count-to-infinity" problem (see chapter 3.3.4).

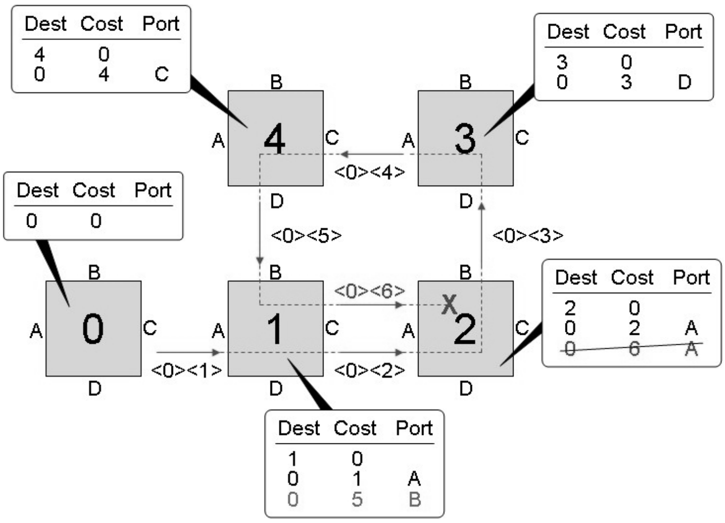


Figure 4.7: Circle of the routing message

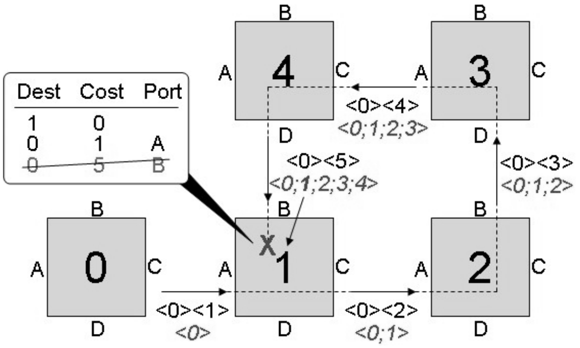


Figure 4.8: Tracking of tokens and stopping of circles

Previously, it was assumed that the costs per module passage always had the value "1". In a real system, however, it has to be assumed that the time for a CU to pass a module can vary depending on acceleration or non-square module geometry. In this case, the sending module will add the time required from the decision point to the module boundary and the receiving module will add the time required from its own module boundary to its own decision point. These times, however, have to be manually programmed into the modules at the time of manufacturing. Alternatively, the system can calculate the costs of this route itself, if the transit times of previous CUs have been measured and the routing tables have been amended accordingly.

Updating of the routing table

Basically, two possible types of layout changes can be differentiated. On one hand, one or more modules can be added; on the other hand, one or more modules can fail and be removed from the system. In both cases, the system has to react to the change.

When a module is added to the system, it sends, like all other modules, its routing information at random intervals to its neighbor modules. When a neighbor module receives the message, it will recognize the new module, activate the port and send its own routing information back. Furthermore, the integration information of the new module is sent to all other modules during the routine update of the system.

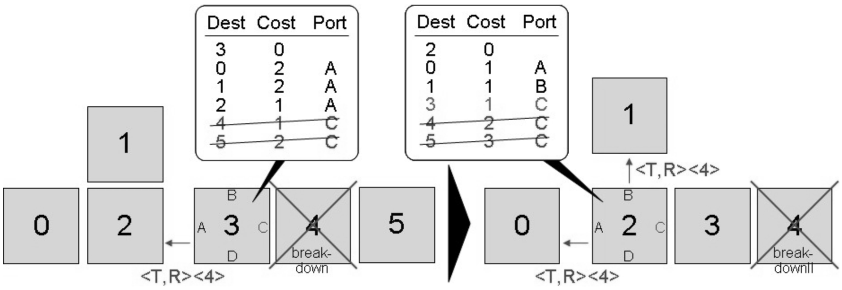
When a module is removed or has failed, the remaining modules react automatically by updating their routing tables. It should be avoided, especially when deliberately removing a module, that a CU is on its way to or even on the module, because this could cause a blockage in the system.

The following procedure will result in an automatic update of the routing table in case of failure or removal of a module:

When a module receives a topology token for the first time from its neighbor module, the receiving port is set to ”active” and the receiving module saves the time of receipt of the data. Because all modules have to send their entries to neighbor modules within a maximum defined time limit (e.g., 10 seconds), the receiving module can assume that it will receive a minimum of one other topology token within this maximum time limit. If this time limit has passed, the receiving module recognizes the failure of its neighbor module and resets the port to ”passive”. In this case, the module deletes all entries in its routing table that run via this port. Furthermore, it sends a reset token to inform other modules of this failure. These then check their own entries to see if they are dependent on the failed module and delete them immediately.

At the time of generating a routing table it is determined through which modules an entry has already passed. This information can be used to decide if an entry is affected by the failure of the module. By updating on a regular basis, the topology tokens disperse alternative routes. This way the system can ”heal” itself (see Figure 4.9).

Modules remember the module ID of a failed module for 5 seconds to prevent the reset token being sent endlessly through the system. During this time, multiple reset tokensthat are received are not sent further.



Dest	Cost	Port
3	0	
0	2	A
1	2	A
2	1	A
~~4~~	~~1~~	~~C~~
~~5~~	~~2~~	~~C~~

Dest	Cost	Port
2	0	
0	1	A
1	1	B
3	1	C
~~4~~	~~2~~	~~C~~
~~5~~	~~3~~	~~C~~

Figure 4.9: Deleting entries after module failure

4.2.2 Identification of the conveyor unit

In a decentralized system, every module can recognize a newly arriving CU and forward it in the direction of the sink module, without endangering the system stability. It can be assumed that a CU always enters a module through one of its ports and that no other CU is located within these module boundaries at the same time (see 4.1.2).
In this dissertation it is assumed that the module has sensors at each of its ports which report the arrival of a CU. As soon as this message has been received, the module is switched on to forward the CU in the direction of the decision point. At the same time the identification system is activated, which reads the CU ID and Dest ID as soon as the CU enters into the reading area. The module records this first contact and blocks itself for the clearance for other CUs. This assures that two CUs never arrive at the same module simultaneously.
If the identification system is not able to read a CU immediately, it will be slowly transported towards the opposite port of the module. Once the sensor registers the arrival of the CU at the opposite module boundary, the actuator is switched so that the CU is transported back towards the center, i.e., the decision point (see Figure 4.10).

The CU is thereby transported backwards and forwards over the identification system until positivly identified or until a predefined number of tries have been unsuccessful. In this case, the CU is forwarded in a pre-selected direction (”No-read” Direction) without a reservation. The ”No-read directions” should be positioned in the layout of the modules in that way that a CU with corrupt identification data will eventually be transported to a predefined place.
In order to guarantee a continuous material flow, the CU should to be identi-

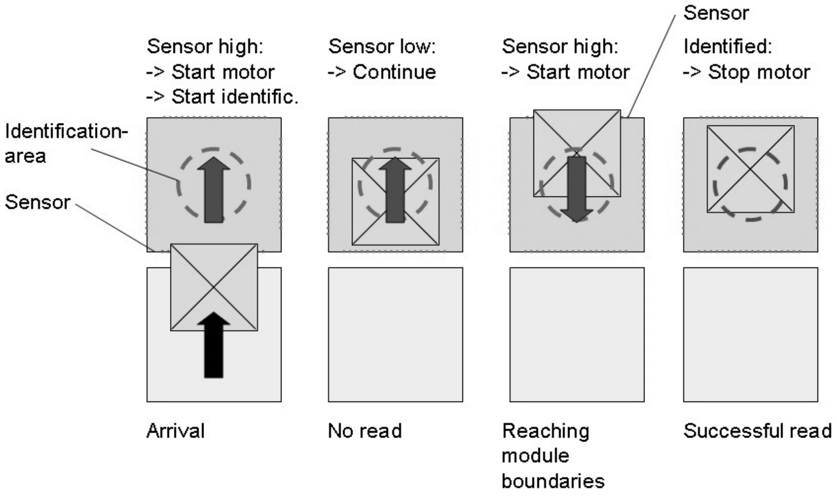


Figure 4.10: CU identification at arrival

fied prior to arriving at the decision point in the middle of the module and the routing finalized. For reasons of symmetry it is assumed that the information is tagged in the middle of the CU. The maximum range of the identification system should not exceed the module boundaries so that CUs cannot be read by neighbor modules. Hence, the time available for the identification procedure is the time the CU needs to go from the module's boundary to its center. In the case of a square module this is the maximum time required to travel half the module's length.

4.2.3 Routing and route reservation

Previous material flow layouts are mostly based on a one-way street principle, in which all conveyor sections have a pre-selected direction, preventing any kind of deadlock. The reasons are missing control algorithms and a loss in throughput during the change of direction. (During the change of direction, no CU can be sent in either of the two directions). MFS controls are only equipped with static right-of-way rules, which guarantee the lowest transit times in accordance with the planned capacity.
Due to the increasing flexibility requirements, more and more situations occur in which it would be an advantage to use a conveyor section in both directions. One example is modern distribution centers, in which truck docks are used either for incoming or outgoing goods, depending on the time of the day.

Less frequently used conveyor sections to long-term warehouses or assembly work stations could also be used more efficiently in times of lower capacity rates by running them in both directions rather than installing two parallel conveyor sections or even completely abolishing continuous conveyors.
One important feature of the new control system is that all conveyor sections can be run in both directions and the routing process is configured to avoid collisions when transporting CUs in opposite directions. This may result in lower throughput, which is discussed further in chapter 4.4. Avoiding collisions in material flow is absolutely essential as opposed to the transport of information packages in IT networks.

The following basic logics can be distinguished for the routing:

- Simple transport of the CU to the port with the best route
- Reservation of the route up to the next node
- Continuous reservation from source module to sink module
- Time-discrete reservation

Being a fully decentralized system, no higher level infrastructure is available that supplies the modules with information about the location of CUs in the system. To avoid collisions, however, it has to be ensured that two CUs on opposing routes will not meet at any time. Therefore it is necessary to reserve the conveyor section to the next node in the desired direction prior to sending the CU so that during this time no CU can be transported in the opposite direction on this section. For this reason the first basic logic (see above list) will not be pursued further for the developed control system.
Furthermore, the modules are not able to buffer any CUs until an opposing CU has passed. Although collisions are no longer possible due to the reservation of the conveyor section, deadlock situations can still occur at those nodes at which two opposing CUs can not pass each other and therefore block the node. This is shown in Figure 4.11, where two CUs reserved the conveyor sections to module 4 and to module 8 respectively, and have advanced to the nodes. But now the CUs can't pass each other because there is only one connecting conveyor section between module 4 and module 8. This is a deadlock situation. For this reason, the second basic logic (see above list) won't be pursued any further, as well.

The danger of a deadlock can be minimized in the developed routing algorithm by reserving the entire conveyor route from the source module to the sink module, before sending the CU. However, the reservation only affects

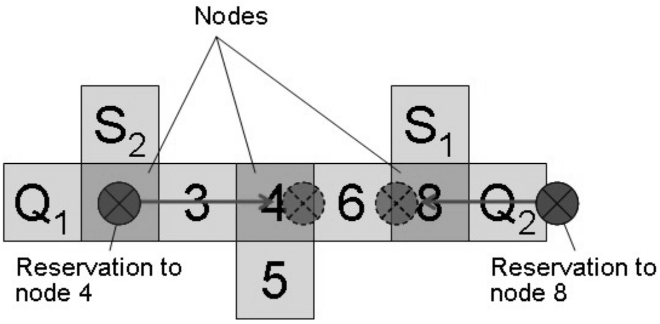


Figure 4.11: Deadlock danger when reserving a partial conveyor route

the blocking of conveyor section of those CUs that are traveling on opposite routes. CUs that cross the conveyor section or travel in the same direction are not affected.

The disadvantage of such a reservation is the possiblly unnecessary waiting time for opposing CUs, which might only use the blocked conveyor section for a short distance. Because a small increase of the waiting time in most practical layouts is not important and the algorithm can be kept quite simple, this logic is discussed further in this dissertation.

The last logic in above list, a time-discrete reservation, is also not discussed further, because the computing time and the necessary amount of data required by the modules would be very voluminous (see "time-discrete module reservation").

Process of route reservation

When a CU is injected into the system, the source module sends out a reservation token in the direction with the lowest costs selected from the routing table. This reservation token contains the module ID of the sink module (Dest ID), the CU ID, and the module ID of the source module. When a neighbor module receives this reservation token, it checks if the Dest ID corresponds with its own module ID. If this is not the case, the request is saved in a reservation table and the reservation token is forwarded in the shortest direction towards the sink module. This port is then blocked for incoming reservation tokens so that no reservations can be made in the opposite direction. The reservation token runs along this route through the CS until it arrives at the sink module. On arrival, the module compares, like all other modules before, the Dest ID of the reservation token with its own module ID. Because the two IDs correspond with each other, the module swaps the module ID of the source module with the Dest ID of the reservation token, attaches clearance information and sends it back to the source module. The route is only blocked for requests in the opposing direction; CUs running in the same direction can still be reserved and forwarded (see Figure 4.12).

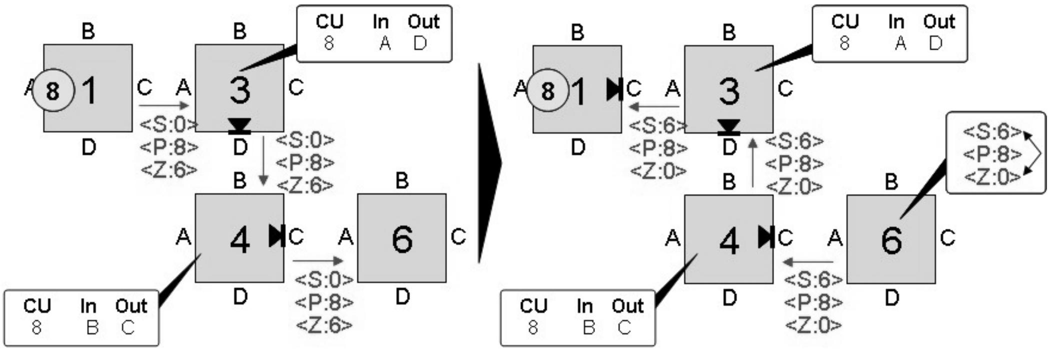


Figure 4.12: Sending of tokens for CU reservation

Furthermore, similar to the topology generation, the reservation token also contains the module IDs of the modules already passed, in order to prevent the reservation token from running in circles.
When the reservation token has arrived back at the source module, the route has been fully reserved and the CU can be sent. Prior to the physical transport, two further characteristics have to be checked to prevent a collision: Firstly, the next module in line has to be asked if it is available to receive and forward a CU and secondly, it has to be ensured that the forwarding of a CU does not lead to a deadlock of the whole system. These two characteristics will be discussed in the following chapters.

In the case that another CU enters the system whose route partially runs in opposite direction (see CU no. 9, Figure 4.13), a reservation token will be sent from the source module as before. The reservation token initially runs smoothly through module 4 because the blocked port does not correspond with the entry port of the reservation token. However, this is the case in module 3. Because the blocked port is being addressed, module 3 deletes the reservation token and sends back a message that this route is barred (x-token). When module 4 receives the x-token, it searches for alternative routes in its routing table. If it cannot find an alternative route, module 4 also deletes the reservation and sends the x-token to the previous module. If the x-token arrives back at the source module, it waits a short while (e.g., 2 seconds) and tries again. Furthermore, module 4 remembers the request from the source

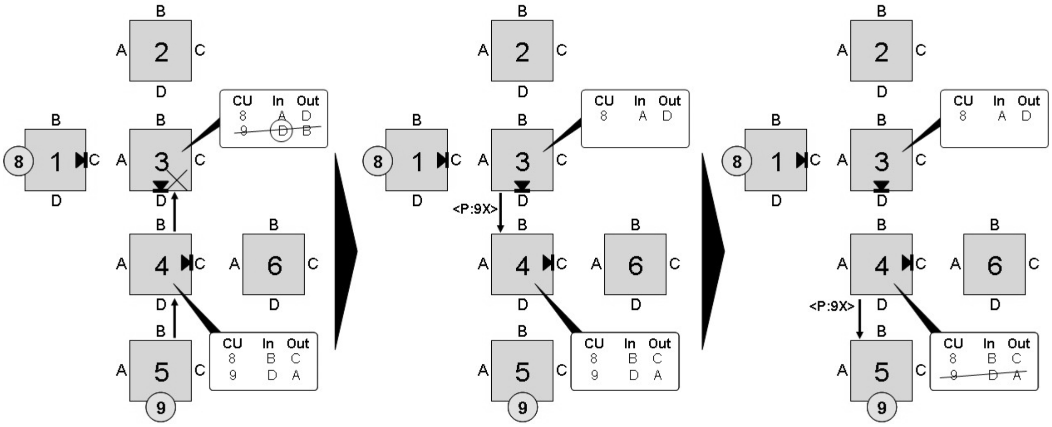


Figure 4.13: Route blockage in case of danger of collision

module and sends a clearance token once the first CU has exited module 4.

If on the way back from a failed reservation a new alternative route is found in the routing table (see Figure 4.14), a new reservation token is sent on the alternative route prior to further sending back the x-token. If the reservation token eventually arrives at the sink module, the CU is sent on this alternative route instead. However, it is not checked how much longer the alternative route is in comparison with the initial route or if it is worth waiting for the clearance of the blocked conveyor section. This possible solution is very complex from a decentralized point of view because the current position of the opposing CU would need to be calculated in addition to the section reservation.

The reservation request will not run in circles on the alternative route because the module ID of all passed modules is attached to the reservation token. If a module receives a reservation token twice, the route will also be blocked and an x-token will be sent back. This prevents reservation tokens or CUs from running in circles. Figure 4.15 shows such a scenario.

At module 13, the conveyor section is blocked for the CU no. 2 by an opposed CU. Module 5, however, determines alternative routes and sends the token towards module 4. This module determines the shortest alternative route via port A, which causes the reservation token to run in a circle and to arrive back at module 5. This alternative route is the blocked as well. The reservation token is sent back to module 4 and finally arrives at the sink module via a third alternative route (port B and module 3). Consequently, CU no. 2 will

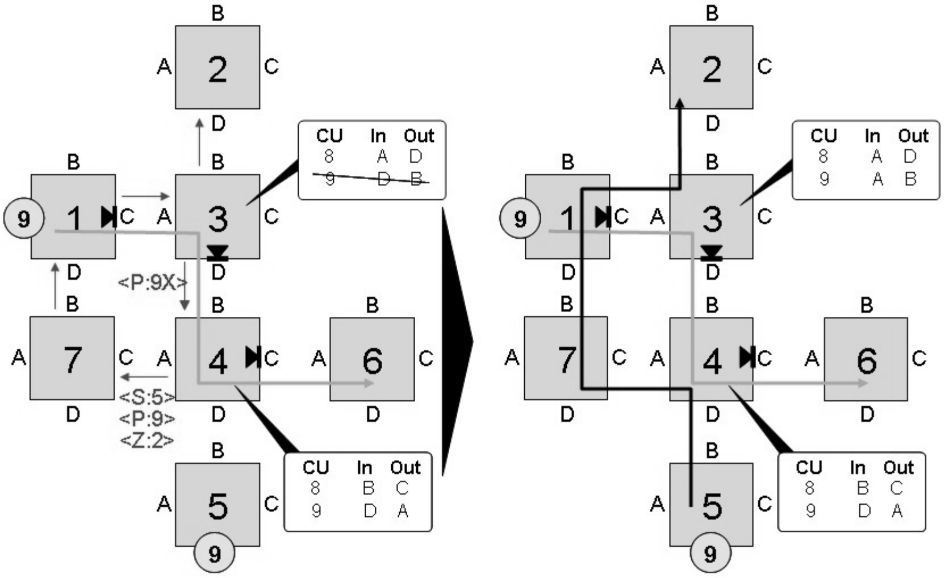


Figure 4.14: Search for an alternative route

travel via module 5, 4, 3 and so on.

Time-discrete module reservation

For an optimal use of the capacity of existing module resources, a time-discrete reservation of affected modules could be a solution, rather than permanently reserving a route between the source module and the sink module. In this case, each module would only be reserved for a certain amount of time within which the arrival of the CU would be expected. A deadlock situation, like during the *reservation to the next node* , is avoided by reserving the entire route up to the sink module prior to sending the CU. However, it is possible that CUs use a "blocked" section temporarily as long as the two time windows don't overlap.

Figure 4.16 shows a situation where CU 1 was injected into the system just before CU 2. CU 1 needs to go to module 9, CU 2 needs to be transported to module 5. Without a time-discrete reservation, CU 2 needs to wait until CU 1 has passed module 7, because the section between modules 6 and 7 has already been reserved for CU 1.
In a time-discrete process, however, module 6 doesn't expect CU 1 until the time $t_1 = 6$ and therefore could give CU 2 clearance as it would pass through at time $t_2 = 3$. In this case it was assumed that CU 2 must not exceed a

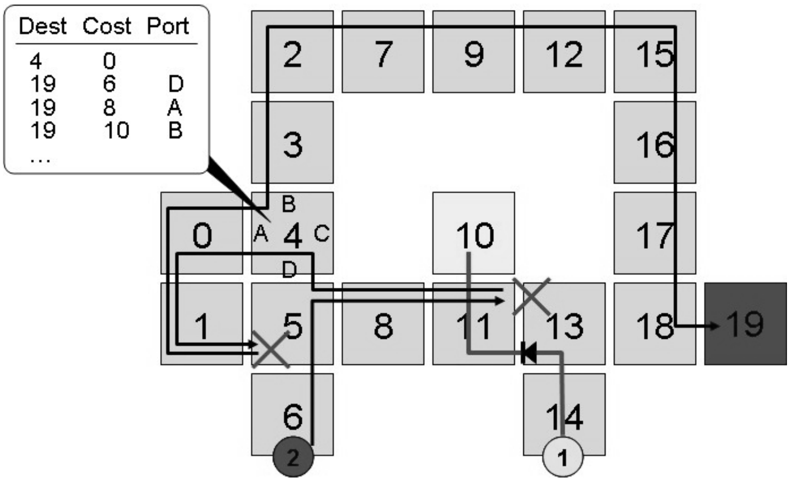


Figure 4.15: Avoidance of a circle on alternative routes

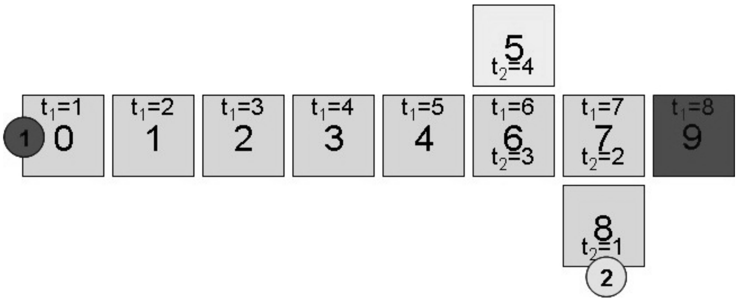


Figure 4.16: Time-discrete reservation of module resources

maximum of two more time units, in order to avoid obstructing CU 1.

In this dissertation, the time-discrete reservation of modules will not be discussed further, because with several CUs in the system, the possibility is very high that expected arrival times cannot be guaranteed due to mutual obstructions of CUs at intersections and therefore time windows might be shifted. In order to avoid deadlock situations, modules would need to permanently adjust their time windows to the real arrival times and update reservations. This would increase considerably the computing time and data transfer, which would restrict the scalability of the system. However, a time-discrete reservation could be of advantage in certain systems and therefore should be analyzed further in future research projects.

4.2.4 Transportation of conveyor units

Shortly after the source module has reserved the route for a CU, the transport along the route begins. During the reservation process, all modules along the route have saved the CU ID together with the entry and exit ports. Hence, no further routing processes have to be run. The modules forward the CU to the saved port immediately after identification. This process saves time and unnecessary data transfer. In case of a technical failure of a module, all reservations are deleted and renewed, so that the CU can use an alternative route as early as possible. However, a failure can bear the risk that no alternative routes are available, which can result in a deadlock. This depends on the system's workload and how many alternative routes exist.
Prior to sending a CU to the neighbor module, several situations have to be checked in order to maintain the efficiency and functionality of the system. Despite having reserved the route, blockages and deadlocks may occur due to the interaction of several CUs in the system. To avoid such situations three requests are made prior to sending the CU:

- Is the neighbor module ready to take a CU?
- Is there a risk of a deadlock situation?
- Is there a risk of system overload?

While this chapter discusses the first request, the following two chapters will discuss the other two requests.

Clearance for the transport of CUs
Prior to forwarding a CU to a neighbor module it needs to be checked if this module is ready to take the CU. This is confirmed with a clearance token that checks the status of the neighbor module. The CU will be forwarded if a positive message is received. If the message is negative, the module will wait a predefined period of time and will then try again.
An occupied module saves the received clearance tokens in order to shorten the waiting times and sends a clearance message to adjacent requesting modules after having forwarded its own CU. These direct clearance approvals interrupt the waiting time until the next request and increase capacity. By alternating the sequence of cleared ports, it can be guaranteed that all neighbor modules are equitably cleared to forward their CUs to the now available module.
At this stage, a prioritization of CUs could be possible if the clearance token contains an appropriate message and if the prioritized CU receives the clear-

ance message from the next available module before any other CU.

4.2.5 Deadlock avoidance

(Tanenbaum 2002) describes the existence of a deadlock in information technology as follows:

"A set of processes is deadlocked when every process in the set is waiting for a resource that must be released by another process in the set"

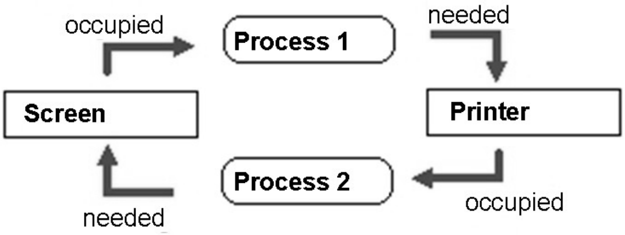


Figure 4.17: Example of a deadlock of two processes, Source: Tanenbaum 2002

A deadlocked system must have at least two processes. Figure 4.17 shows such an example. Process 1 occupies the screen while it is waiting for the printer. At the same time, process 2 occupies the printer while it is waiting for the screen.
A more common example is found in road traffic when four cars arrive at an intersection at the same time with no rule for the right of way. In the case of "right before left", all cars are waiting for the right car to go first and nobody will move (Peterson and Silberschatz 1983).

The phenomenon of deadlock has been studied extensively in the context of computer operating systems (Panson 1985), (Deitel 1983). It is well known that the following four conditions are necessary for a deadlock to occur among concurrent processes (Coffman, Elphick, and Shoshani 1971), (Coffman and Denning 1973):

1. **Mutual exclusion:** processes require the exclusive use of resources

2. **Hold while waiting:** processes hold onto resources while waiting for additional required resources to become available
3. **No pre-emption:** processes holding resources determine when they are released
4. **Circular waiting:** closed chain of processes in which each process is waiting for a resource held by the next process in the chain

To avoid deadlocks, it is sufficient to assure that at least one of these four necessary conditions is not fulfilled. For the modules of the decentralized material flow system (MFS) considered in this dissertation, the first three conditions are always present: resources (modules) can handle only one CU at a time, modules hold onto CUs while waiting for the next module specified in the transportation route, and a module occupied with one CU cannot be pre-empted by another module.
Thus, to avoid deadlocks, the focus is to ensure that the fourth condition (circular waiting) will never occur.
Although it is easy to avoid circular waiting among the modules waiting for their neighbor, a good deadlock avoidance algorithm should allow the maximum use of resources to optimize the throughput of the MFS. For example, the easiest way to prevent the circular waiting condition is to allow only one CU in the system at a time. Such a trivial policy is, however, overly conservative. The algorithm presented in this dissertation will allow a maximum possible load of CUs within the system without risking a deadlock.

Analysis of critical module states
Figure 4.18 shows the different states of a module from the appearance of a new CU up to the transfer to the next module on a predetermined (reserved) route to the destination.

It shows that the following four critical states can occur:

1. CU can not be identified
2. No route to the destination is available in routing table
3. Never receives a successful reservation
4. Never receives a clearance token

The first two states can occur due to a technical failure, misuse or insufficient mechanical design of the modules. They are not described further in this

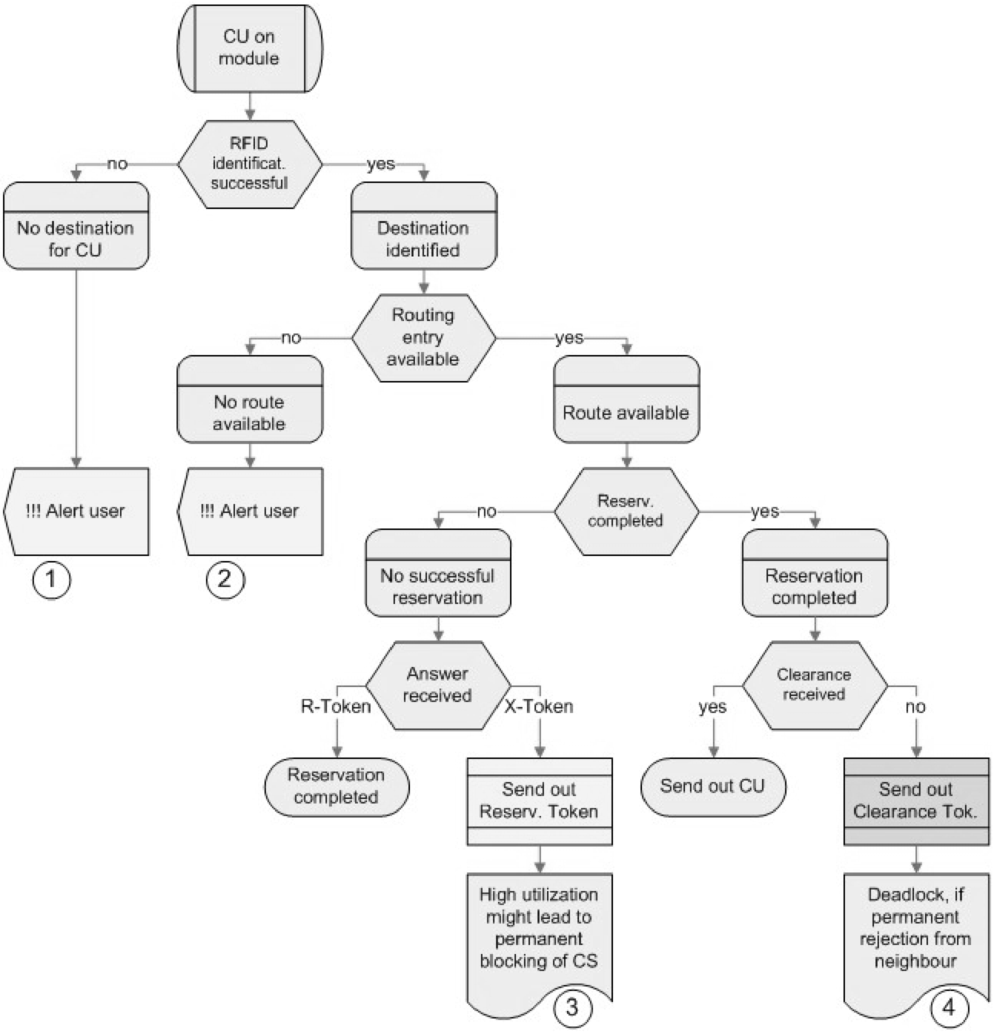


Figure 4.18: State diagram of a module which carries a CU

dissertation as they are not connected to any deadlock situation.

The third state can easily occur during a high utilization of the CS, if a conveyor section is the only connection between several sources and sinks, and are to be used in both directions (see Figure 4.19). To ensure a maximum acceptable waiting time for the CU, the layout must be changed or additional prioritization algorithms must be implemented (subject of ongoing investigations at the IFL). As the blockages dissolve autonomously by reducing the utilization, the fourth condition of "circular waiting" is not fulfilled and a deadlock is not identified.

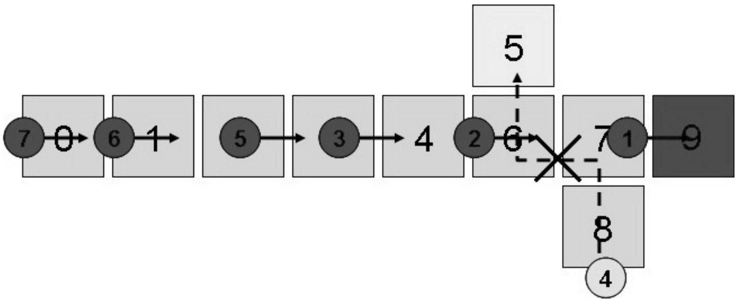


Figure 4.19: Example for condition three, module 8 gets no valid reservation due to a high utilization of the CS

The last critical state turns into a deadlock if the wait for clearance from the next module turns into a circular waiting, because both modules are part of a circular chain in which all modules are occupied with a CU and unable to give clearance. To avoid such a situation, the following model is created, which is supported by axioms:

Basic model for the avoidance of a deadlock
The distributed system consists of a finite set of modules. A module (M) is in one of three general states: **active**, **requested** or **blocked**, and has a finite number of four ports (P) that are able to receive CUs and requests (rq).

Description of module states

S1 M is **active**, if it carries no CU and was not requested (rq) to receive one yet

S2 M is **requested**, if it has received a rq but has not sent out clearance (cl) for the handover yet

S3 M is **blocked**, if it has sent out clearance cl to receive a CU or if it already carries a CU as shown in the state diagram 4.18

With the description of several module process axioms, the basic logic for M is described to ensure a deadlock-free material flow system:

Initial module process axioms

A1 When M_i wants to move CU_i to M_j it sends rq_i to M_j

A2 If M_j is blocked it ignores the request

A3 If M_j is not blocked it sends a so-called "deadlock token"($dt_{i,j}$) to M_k, which is the next module on the predetermined route of CU_i and remembers the port number (P_i) of the incoming request rq_i

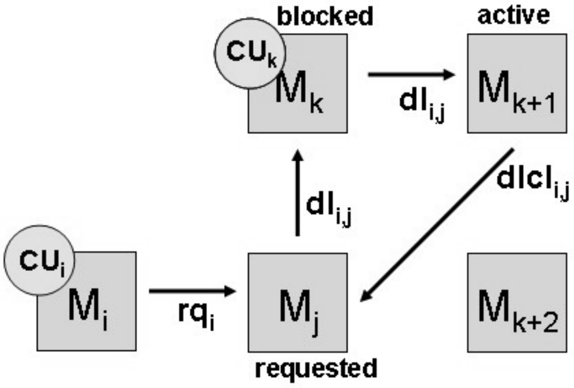


Figure 4.20: Deadlock avoidance process

The behavior of a network of modules in terms of the three different states is described as follows:

Module process axioms

A4 If M_k receives $dt_{i,j}$ and is **active**, it sends $dt_{i,j}$ back to M_j including a "Deadlock-Clearance" ($dtcl_{i,j}$)

A5 If M_k receives $dt_{i,j}$ and is **blocked** or **requested**, it sends $dt_{i,j}$ to the next module M_{k+1} on the determined route of **its own** CU_k

A6 If M_j receives $dt_{i,j}$ from a port other than P_i, deadlock danger is recognized and $dt_{i,j}$ is ignored

A7 If M_j receives $dt_{i,j}$ from port P_i it sends out clearance to M_i and becomes **blocked**

A8 If M_j receives $dtcl_{i,j}$ it sends out clearance to M_i

A9 If M_i receives cl_i it moves CU_i to M_j and returns to an **active** state after a certain time (after CU_i has entirely left M_i)

A10 If M_{sink} receives $dt_{i,j}$ it sends out $dtcl_{i,j}$ to M_j

These axioms are sufficient to explain the whole process for deadlock avoidance.

Proof of correctness

If a system consists of a finite set of modules, of which some are arranged in a circle, there is danger of a deadlock. As long as there is at least one module in an **active** state, the last condition (circular waiting) is not fulfilled (see [A4] and [A8]), and movement is possible.

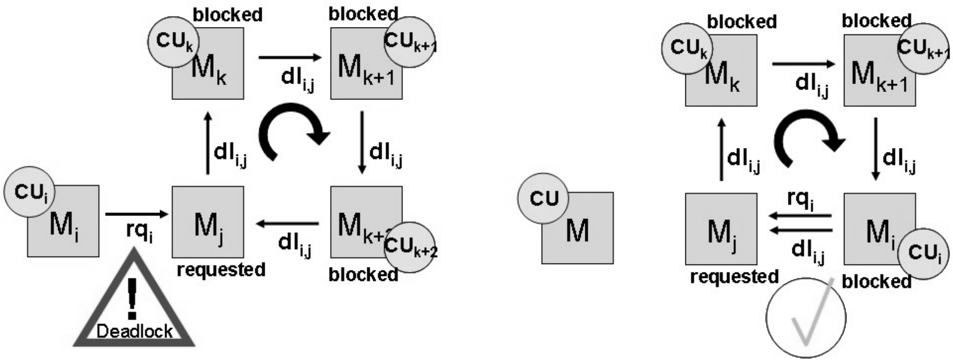


Figure 4.21: Identification of circle member by comparison of receiving ports

A module can only turn into a **blocked** state if another module in the circle is **active** ([A4] and [A8]) or if it receives its transmitted deadlock token *dt* from the same port from which *rq* was received ([A7]). This is only the case if the requesting module M_i is part of the circle itself ([A5]). Although a clearance *cl* from M_j to M_i will **block** M_j ([S3]), the process will re-**activate** module M_i, which is part of the circle, after a certain time ([A9]) and the last condition (circular waiting) is not fulfilled.

A simulation of the system performance (Figure 4.22) in a circle (like Figure 4.23 a)) shows the impact of a deadlock request. The throughput per minute was evaluated with a gradual increase of the input. It shows clearly that without a deadlock request, a deadlock situation occurs once a certain injection rate is reached and the material flow ceases. With a deadlock request, however, the throughput decreases when a certain injection rate is reached, but a blockage will never occur. The decrease of the throughput is discussed further in detail in chapter 4.4.

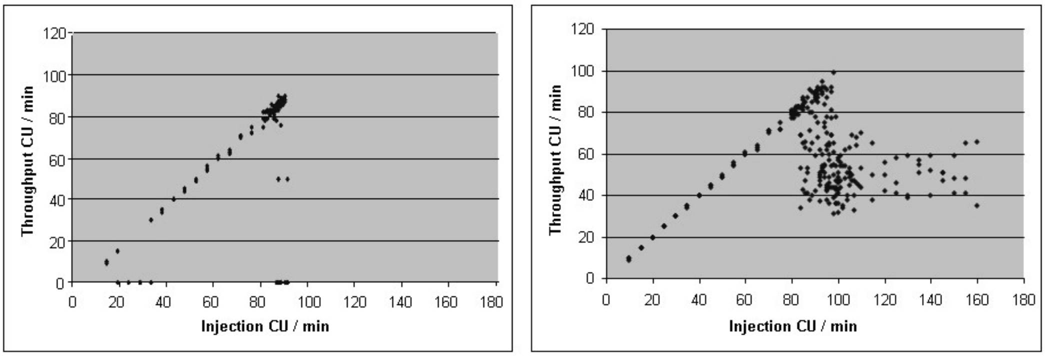


Figure 4.22: CS throughput; left: without deadlock check, right: with deadlock check

Examining different topologies further in regard to deadlocks, three different topology types unfold which are differentiated as follows:

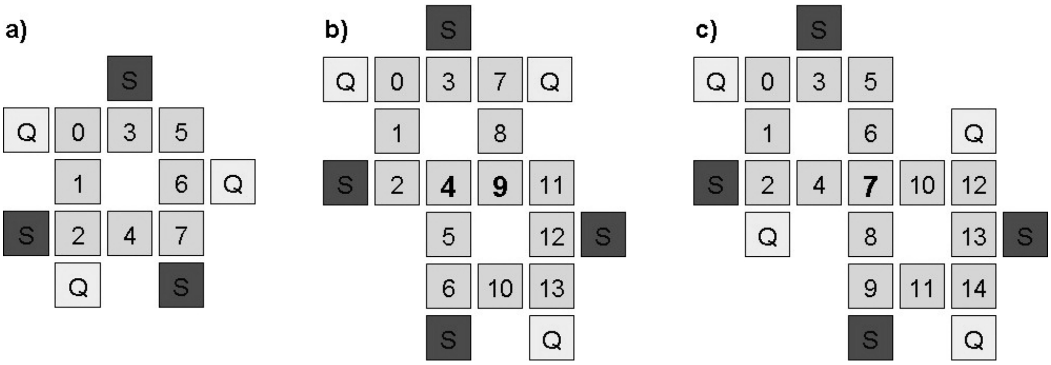


Figure 4.23: Circle topologies: a) independent circle; b) circles overlap in several modules; c) circles overlap in one module

- Most common are module layouts with individual circles, that are completely independent from each other (a)
- Secondly, circles can be connected through several modules (b)
- Finally, there are layouts in which several circles join in one module. In this case, the module is the only connection between the circles (c)

From these three main types, all deadlock-endangered layouts can be designed and described.

a) Independent circle

For the independent circle, deadlocks can be avoided by applying the above-discussed axioms. Thereby it doesn't matter how many modules M are in the circle or how high the throughput is.

Deadlocks can be avoided even if several CUs want to snap up the last remaining free spaces within a circle. In this case, the modules of the circle switch to the status **requested** immediately after the request (rq) has been received ([S2]) and the deadlock tokens are forwarded within the circle ([A5]). If two CUs put in their requests to enter the circle at the same time, neither of the two receive the clearance; therefore, the system would remain fully functional even with only two non-blocked modules.

b) Circles which overlap over several modules

In this case, the deadlock problem is resolved with the same algorithm as in the independent circle. The transmission of the deadlock token prevents a closed circle of CUs from developing, which blocks itself.

If two circles overlap over several modules (see Figure 4.23 b)), this section is being used in the same direction in both circles. Because in a closed circle, it can never happen that all places are taken at the same time due to the deadlock request, and a minimum of one place is always free. A worst-case scenario would be if this free place were in the first module connecting the two circles, because when a CU makes its next move, one of the two circles would be closed (see Figure 4.24 a)). However, a deadlock situation only occurs if the last joint module (modules with CU 10) of the circles carries a CU which is going to be transported in the same circle. This is prevented by the deadlock request, because then, the CU would not receive clearance and the CU of the other circle would be allowed to move.

Hence, when both modules with CU 3 and CU 6 send the request token rq to module 4, for each CU a deadlock token will be released. The process of a deadlock request depends on the next module on the route of CU 10, because deadlock tokens are always forwarded in the same direction as its own CU ([A5]). This CU also decides which circle is going to be closed, because both circles cannot be closed at the same time. As the module carrying CU 10 forwards the deadlock token to the upper circle, CU 3 gets clearance, whereas CU 6 has to wait (see Figure 4.24 b)). Although the lower circle is working to full capacity, CU 10 will leave the circle upwards as soon as the empty gap of

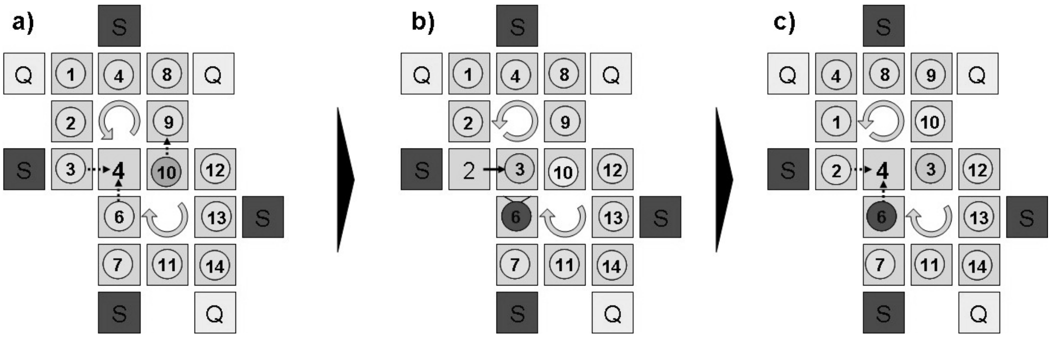


Figure 4.24: Process of deadlock prevention in overlapping circles

module 2 has arrived at CU 10 after CU 9. Although this process has finished when CU 3 has been transported to its right, the same process begins all over again with CU 2 and CU 6 (see Figure 4.24 c)). This time it depends on the ongoing route of CU 3.

c) Circles which overlap in one module

The avoidance of a deadlock as described in the general definition is also assured in this scenario. If one or both of the circles are almost fully **blocked** with CUs, the above-mentioned algorithm will not allow the last module to receive another CU. Consequently, the last condition "circular waiting" will never be fulfilled.
Unfortunately, in high utilization, the described deadlock algorithm could lead to a new type of system halt, which will be called **cross deadlock**.

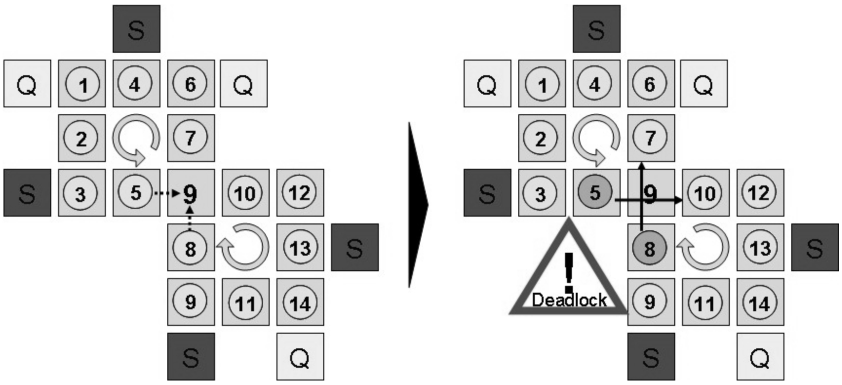


Figure 4.25: Cross deadlock, if two circles overlap in one module

A cross deadlock can occur when all modules in the two circles are completely **blocked** except the module were the two circles overlap (see Figure 4.25).

In this case, if in each circle the CU (5 and 8) in front of the overlapping module (9) is on the way to enter the opposite circle, the system breaks down. This is because when the overlapping module (9) sends out a deadlock token around the opposite circle, for both CUs, it will recognize the danger of a deadlock. Consequently, both requests will be ignored and the conveyor system will come to a halt.

To find a prevention against cross deadlocks, the condition of ”circular waiting” must be prevented not only within single modules but also within the next level of aggregation which are interacting sub-systems as, for example, two circular layouts connected over one module. In the example of Figure 4.23, c), each circle can be seen as such a sub-system that interacts with the other over the connecting module (module 7). Refering to the original example for deadlocks (see Figure 4.17) in case of a deadlock, each sub-system can be seen as a process that needs to use a resource of the other process by handing over a CU. At the same time, each sub-system is not able to take another CU as only one free space in the circle is left. This can only occur when the sub-systems are connected over one single module because if the sub-systems are connected over several modules, the last connecting module can not act for both sub-systems as it carries a CU that is dedicated to one of the two sub-systems. This means, that at least one of the two sub-systems is not waiting for the other, as described in ”b) Circles which overlap over several modules”.
However, if only one module connects two subsystems and each subsystem is waiting for the other, a cross deadlock has occured, which can be avoided as follows:

For modules, which have a maximum of four neighbors, the condition that two CUs are waiting to enter an opposite, almost-blocked circle can occur in two different ways, which are shown in Figure 4.26.

Modules can prevent against cross deadlocks during the process of route reservation before CUs are injected into the system at the source module. When the reservation token is sent out from the source module to the sink, each module on the route that fulfills the necessary conditions to cause a cross deadlock (beeing the only connecting module of two circular sub-systems)

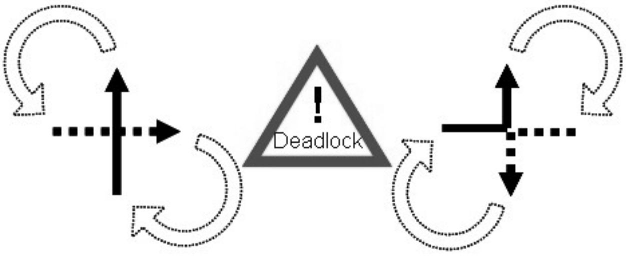


Figure 4.26: System states for cross deadlock; left: 180° request, right 90° request

must check if the requested reservation is secure. If not, the reservation is rejected with an x-token (see chapter 4.2.3) and handled as if it were a blocked route.

Necessary module condition to be able to cause a cross deadlock:

- Each port of the module is connected to another module (4 neighbors)
- Both pairs of ports of the module form (together with other modules) a circle

Cross deadlock check for a 180° request
If the outgoing port of a requested route is on the opposite side of the port from which the request was received, it is called a *180° request*.
If such a request is received, the module checks if it has already reserved another CU perpendicular to it. If this is the case, the request is rejected and an x-token is sent to the requesting module (see Figure 4.26, left).

Cross deadlock check for a 90° request
If the outgoing port of a requested route is on the left or right side of the port from which the request was received it, it is called a *90° request* .
In this case, the module checks if there is already another reservation that came from the opposite port and which follows a route to the opposite output port of the current reservation. Again, if this is the case, the request is rejected and an x-token is sent to the requesting module (see Figure 4.26, right).

Self-detection for cross deadlock danger
Although the above-described process prevents cross deadlocks, there is a disadvantage that should be mentioned. If a conveyor system is running with

high utilization, there is a high probability that a once-rejected request will be rejected for a long time, because the relevant module has a long list of blocking reservations, which continue to exist because of CUs being continuously injected into the system. It might happen, that the requesting module will never succeed, as in the case of the third state in the state diagram (see Figure 4.18). To minimize the waiting time, only modules that are really in danger of a cross deadlock should reject unsecure requests. This is only the case for modules that mainly fulfill the second of the previously mentioned conditions (both pairs of ports of the module form a circle).
To check this, all modules occasionally carry out a self-diagnostic. This is done by a "diagnostic token" that is sent to one of the four connected ports. When such a diagnostic token is received, the module checks its routing table for an alternative route to the transmitting module. In this case, the message will be forwarded in this direction. If it's a circle situation, the message will come back to the sending module. If it receives back the diagnostics token, it sends out another diagnostics token to one of the remaining neighbors. The module repeats this process with all four ports. If it received back the diagnostic token each time from a different port, it recognizes that it has been positioned as the only link between two circles and carries out the cross deadlock check every time it receives a request.

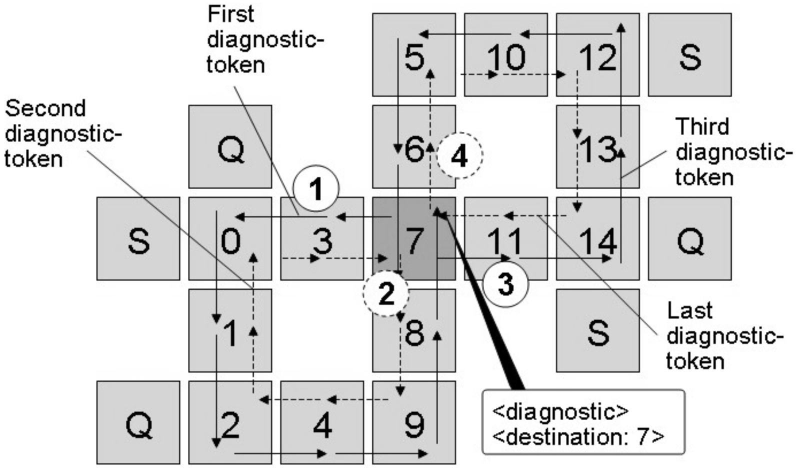


Figure 4.27: Self diagnostics to prevent a deadlock with topology check

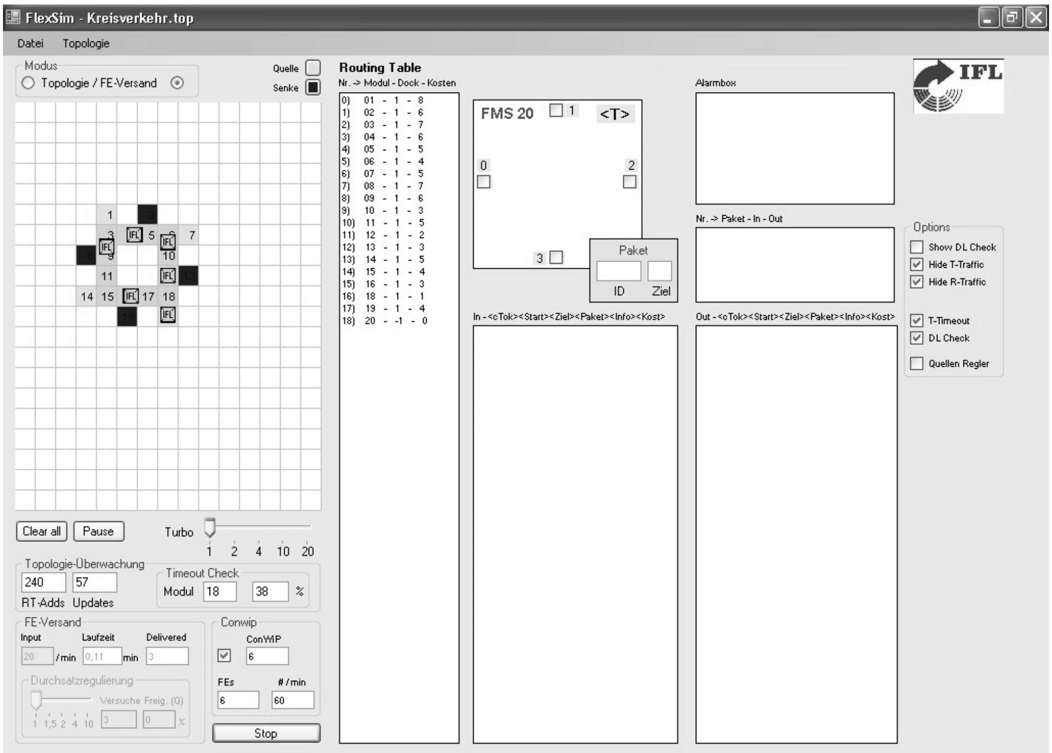
4.3 Throughput analysis

To determine the performance and efficiency of the material flow system with decentralized control, a simulation program was developed and with its help a large number of independent modules can be connected together to form a topology. The modules can be arranged together arbitrarily and CUs can be injected into the newly-created system.
For the throughput analysis, a selection of basic topology forms, as, for example, a straight transport path or a circular path were chosen and examined in the simulation environment. Subsequently, a large number of CUs were sent through the system according to prescribed rules and their throughput behavior was analyzed. The study showed that topologies with circular paths suffer a sharp drop in throughput when the injection rate into the system is too high. The cause for this are the reciprocal blocking effects at intersections. To prevent these drops in throughput, the subsequent chapter 4.4 concerns itself with a self-regulation of the system.

4.3.1 Simulation environment

The simulation environment is based on a Visual Basic application that generates for each module a separate instance, which in turn starts its own thread (independent process). In this thread, the control algorithm of the module will run cyclically. In this way, each module reacts independently to incoming signals, like, for example, the arrival of a token or a CU. Also, a counter runs in each thread that triggers actions after a preset time has passed. This can, for example, be the renewed sending of a transport request or the regular verification of the existence of the neighboring module.
Parallel to the module instances, a user interface is implemented in the simulation environment that visualizes the actions of the individual modules. With this, the entire communication between the modules as well as the physical movement of the CUs from module to module can be observed. With the user interface, modules can be arranged on a surface, and sources and sinks can be defined. As soon as modules are added to the layout, they start the communication and with that the generation of the routing tables. After a few seconds, the modules are ready for the shipment of CUs.

An analysis function's duty is to have several CUs transported from randomly selected sources to randomly selected sinks for a predetermined period of time.



FlexSim - Kreisverkehr.top

Modus: ○ Topologie / FE-Versand ◉

Quelle □
Senke ■

Routing Table
Nr. -> Modul - Dock - Kosten

```
0)   01  -  1  -  8
1)   02  -  1  -  6
2)   03  -  1  -  7
3)   04  -  1  -  6
4)   05  -  1  -  5
5)   06  -  1  -  4
6)   07  -  1  -  5
7)   08  -  1  -  7
8)   09  -  1  -  6
9)   10  -  1  -  3
10)  11  -  1  -  5
11)  12  -  1  -  2
12)  13  -  1  -  3
13)  14  -  1  -  5
14)  15  -  1  -  4
15)  16  -  1  -  3
16)  18  -  1  -  1
17)  19  -  1  -  4
18)  20  -  -1  -  0
```

FMS 20

Paket: ID Ziel

Alarmbox

Nr. -> Paket - In - Out

In - <cTok><Start><Ziel><Paket><Info><Kost>

Out - <cTok><Start><Ziel><Paket><Info><Kost>

Options
- □ Show DL Check
- ☑ Hide T-Traffic
- ☑ Hide R-Traffic
- ☑ T-Timeout
- ☑ DL Check
- □ Quellen Regler

Topologie-Überwachung: RT-Adds 240, Updates 57

Timeout Check: Modul 18, 38 %

FE-Versand: Input 20 /min, Laufzeit 0,11 min, Delivered 3

Durchsatzregulierung: Versuche 3, Freig. (Q) 0 %

Conwip: ☑ ConWIP 6, FEs 6, # / min 60

Figure 4.28: User interface of the simulation environment

The following two modes exist for the analysis function:

First, a defined number of CUs per time period can be injected into the system (theoretical injection). Subsequently, at regular time intervals, how many CUs reached their destination is recorded.
When a CU is injected into the system, the source module is chosen randomly from all free source modules. If the program does not find a free module, then it waits until a source becomes available. As long as the system is not overloaded, new CUs can be allocated to free source modules. When the system starts to become overloaded, CUs that have been injected into the system can no longer be transported onward from the source modules so that there are no free source modules available. Since new CUs can only be injected from free sources, the actual injection rate decreases and therewith the measured throughput.

Second, a CONWIP analysis (**CON**stant **W**ork **I**n **P**rogress) can be conducted. For this, a predefined, constant number of CUs is injected into the system and the resulting throughput is analyzed. As soon as a CU arrives at the sink module, a new CU is injected from an arbitrary, free source. To enable a comparison of different conveyor systems, the number of the CUs in the system is set in proportion to the number of modules in the system. This way, a conclusion can be drawn about the throughput in relation to the filling rate. With a low filling rate there are only a few reciprocal interactions between the individual CUs. The probability of reciprocal interaction increases with the filling rate, however, so that momentary blockages at intersections occur more frequently. This leads to the individual CUs having to stay in the system longer and therefore fewer new CUs per time period pass through the system.

4.3.2 Throughput calculation

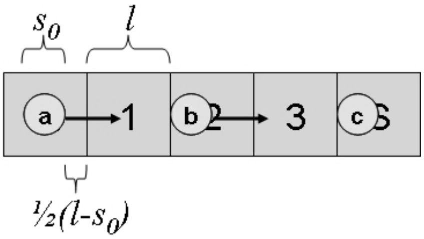


Figure 4.29: Calculation of the maximum throughput

Owing to the basic requirement that only one CU can be within the module's borders at a time, the minimum distance between two CUs $s - s_0$ is at least the module length l. In addition, a CU must wait at the decision point, that is, the middle of the module, until the preceding CU has left the module. Consequently the distance of the CU to the edge of the module is also added to the module length l, which equals $\frac{1}{2}(l - s_0)$.
The maximum throughput γ of a straight conveyor section without idle times, without accelerations, and at a velocity of v is calculated using the following formula:

$$\gamma = \frac{v}{s} = \frac{v}{s_0 + l + \frac{1}{2}(l - s_0)} = \frac{v}{\frac{1}{2}(s_0 + 3l)} \quad (4.1)$$

Since a module needs a certain amount of time to detect the departure of a CU and to send the transport clearance, an idle time t_0 must be added for each CU. It must also be taken into consideration that a CU is identified upon arrival and then the routing is processed. This time is called t_r. Furthermore, the cycle time is increased again by the duration of the deadlock check t_d, which is to be conducted before each transport of a CU. The maximum throughput of a straight conveyor section is calculated therefore as:

$$\gamma = \frac{v}{\frac{1}{2}(s_0+3l)+(t_0+t_r+t_d)v} \quad (4.2)$$

With adequately fast hardware, the value t_0 can be neglected. Moreover, a module that is becoming available sends an appropriate message to its predecessor so that if necessary, it can transport the next CU without delay.

With an early identification via RFID before the decision point and an existing reservation, t_r can be set to 0, since no further route processing must take place.

When there are few congestions in the system, t_d can also be set to 0 since the directly adjoining modules can immediately send a clearance, even before the CU has reached the decision point. When the system starts to become congested, the deadlock check runs through several modules and thus can trigger waiting times in the range of a few seconds. In this case, the CU must be halted at the decision point, which results in further drops in the throughput due to acceleration procedures.

Direction changes, branches and intersections

For a change of direction on a simple conveyor section or a branch, the extra time lost during the switching to the second direction of motion and the acceleration t_s is added to the calculation. Corresponding to the frequency of direction changes, t_s is multiplied by the switch probability ρ to determine the maximum throughput. This leads to the following formula:

$$\gamma = \frac{v}{\frac{1}{2}(s_0+3l)+(t_0+t_r+t_d+\rho t_s)v} \quad (4.3)$$

Throughput drop

If an intersection has several inflow branches, the sum of which lies above the maximum throughput of the intersection, then a congestion of CUs occurs. If one of these inflow branches is connected to the outflow of the intersection to a circular path (see Figure 4.30), then the congestion can run in a circle. At this point, a CU can only be transported into the last free gap in the circle, which leads to a significantly reduced throughput.

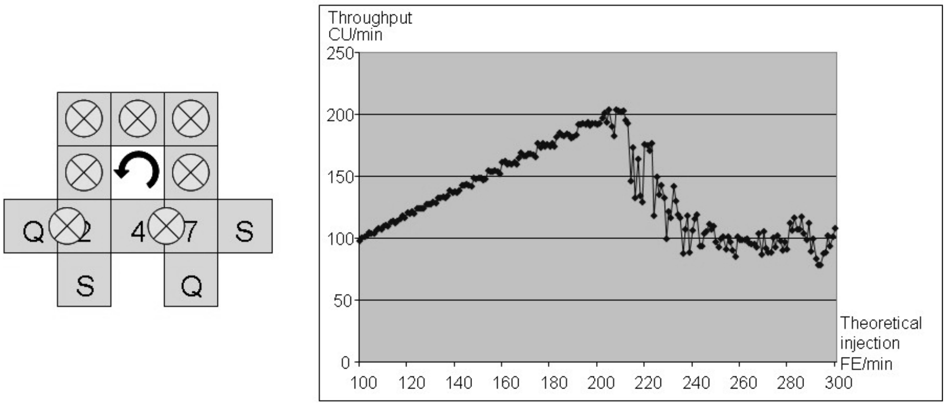


Figure 4.30: Throughput drop when a critical capacity load is exceeded (Circular path)

The deadlock query prevents a complete standstill, but with a high rate of injections into the system, the total throughput of the system lies well below the theoretical maximum throughput of the individual modules. This drop in throughput when a critical capacity load is exceeded occurs only in complex layouts that have circular paths and multiple sources and sinks. Figure 4.22 also shows that the system levels off to a constant throughput even when the theoretical injection rate is increased further. The reason lies in the fact that after the drop in throughput, all sources are occupied with CUs and immediately after they become free, they receive another new CU to inject into the system. The system therefore remains constantly in an overloaded state.

4.3.3 Topology analysis

For the study of the throughput of the decentralized control, various conveyor system layouts were selected and, with the help of the simulation, their throughput behavior was examined. The following different layouts are examined below:

- Simple conveyor line
- Fish bone

- Linear sorter
- Circular traffic
- Two intersecting circles
- Complex topology with several circles

The study of the throughput was conducted using the CONWIP analysis because it offers a good comparison of various topologies. For each filling rate, the average throughput was measured. The respective filling rate is equivalent thereby to the relationship of the number of CUs to the number of modules in the system. For the study, the filling rate was increased in steps from 0 to 100%, whereby 100% is equal to the total number of modules excluding the source modules. For each filling rate, the throughput for approximately 5000 CUs was measured in order to obtain a representative average. In addition, the system was simulated using a velocity four times faster (turbo) than a real implementation to generate more test values in a shorter period of time.

Simple conveyor line

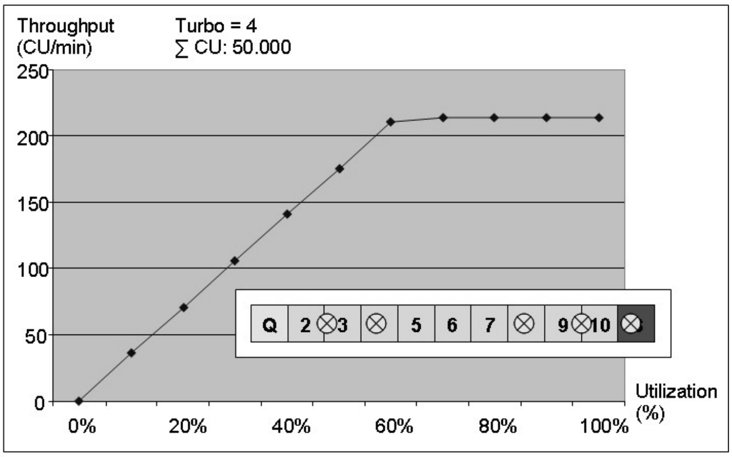


Figure 4.31: CONWIP analysis "simple conveyor line"

The simple conveyor line (see Figure 4.31) is the simplest case for the arrangement of multiple modules. No intersections or branches exist and the maximum number of neighbors is $n = 2$. The challenge for the decentralized control lies here in the generation of the topology at the beginning and in the forwarding of the CUs in the direction of the sink modules. Collisions or deadlocks are not possible.

With the help of the throughput calculation, the maximum throughput of the system can be calculated. For a module length of $l = 0,5m$, a CU length of $S_0 = 0,4m$ and a velocity of $v = 3,5\frac{m}{s}$ (velocity times 4), the calculated maximum throughput - neglecting the idle time, reservation time and deadlock query time - is:

$$\gamma = \frac{v}{\frac{1}{2}(S_0+3L)} = 221\frac{FE}{min}$$

The simulation confirms the value and shows that the maximum throughput is reached at a filling rate of approximately 50%. This filling rate corresponds to the necessary minimum distance between CUs, which was described in the throughput calculation. A further increase in the filling rate no longer results in a change in the throughput, because the CUs on a conveyor line with no branches do not have any interaction with each other. The throughput is thereby defined simply by the velocity at which the CUs are injected into the system at the sources.
For the simple conveyor line, the number of injected CUs consequently does not play a role because even at a high filling rate, no drop in throughput occurs. A regulation of the amount of CUs is therefore not necessary.

Fish bone

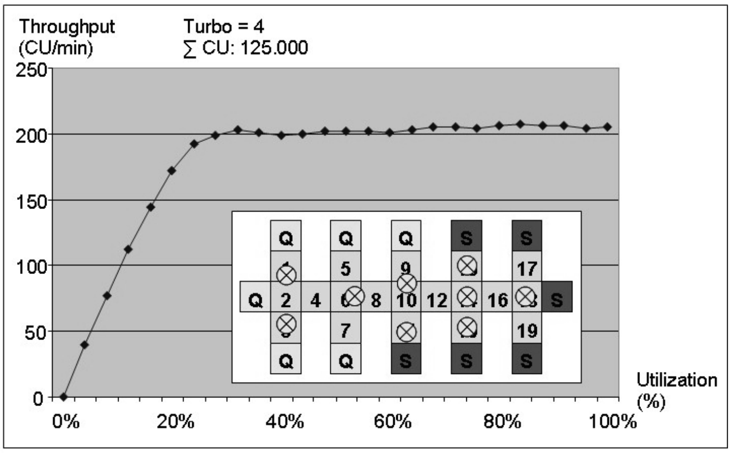


Figure 4.32: CONWIP analysis "fish bone"

In the fish bone layout, several sources are connected to a conveyor line. Then the conveyor line branches off again to multiple sinks. The bottleneck of the

system is the connection between the source area and the sink area, and also those modules that connect multiple sources and multiple sinks together (for instance, module 10 in Figure 4.32).
In this layout, in adition to the generation of the topology, the collision avoidance capability of the decentralized control will be verified. Especially at intersections it is important that the CUs are not sent to the connecting module at the same time. To eliminate waiting times, modules that will be free soon send a message to their predecessors as soon as the modules can take a new CU. The branches do not need an algorithm for collision avoidance, since the CUs are forwarded along the only possible route in accordance with the routing tables.
The analysis shows that the maximum throughput occurs already at about a 25% filling rate. The cause of this is that the system has an especially pronounced bottleneck, which, as opposed to the other system sections, reaches its throughput limits already at a lower filling rate. If one compares this to the maximum throughput, it can be seen that the same number of CUs can be transported as with a simple conveyor line. Here, too, the throughput does not drop and hence a injection regulation is also not necessary here either.
An examination of the filling rate after which an increase in the throughput no longer occurs also provides information about the effectiveness of the layout itself. With a lower optimum filling rate, it can be assumed that several route sections tend to be underutilized, whereas other route sections tend to be over utilized. If the efficiency of such a system is to be improved, then consideration should be given to adapting the bottleneck to the throughput of the rest of the system through an alternative layout or by an expansion of the modules.

Linear sorter

The linear sorter (see Figure 4.33) consists of multiple sources and sinks that are connected to each other through a common conveyor route. This topology can also be compared to a bus system in network technology, where all clients are connected to each other with a cable. The challenge for the control is the coordination of the individual CUs on a shared line.
Because there is only one possible route from each source to each sink, a CU must wait whenever the route section was already reserved for an oncoming CU. This layout is one of the simplest cases in which the reservation logic comes into use. In this case, the entire route from the source to the sink is reserved before the transport starts.

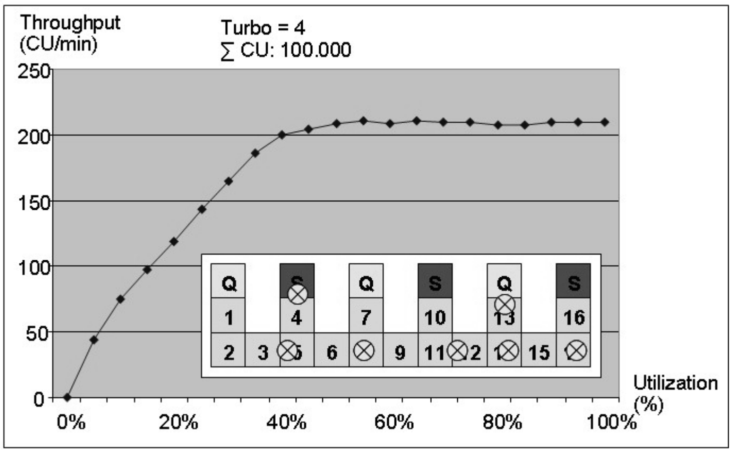


Figure 4.33: CONWIP analysis "linear sorter"

The analysis shows that also with this conveyor system few blocking effects occur and the maximum throughput has a similar magnitude as with the preceding systems. However, with this layout long waiting times sometimes occur for individual CUs until its route section becomes available, because with a high injection rate, the probability is small that a section that is already reserved for oncoming traffic will become free again.
The maximum throughput with this layout is reached at a filling rate of approximately 50%, which speaks for an efficient utilization of the available modules. Also, the throughput rate shows a value comparable to the previous systems, so long as one ignores the fact that some CUs wait very long until the route is free and therefore in some cases very long transport times arise.

Circular traffic

The circular layout (see Figure 4.38) is the simplest conveyor system in which a deadlock can occur. The distinctive feature of this layout is also that a CU has multiple possibilities to reach a sink. Hence the system must make a routing decision. Depending on which route sections were already reserved for a transport direction, new CUs follow this given direction. If the system has a lower filling rate, then the direction of movement in the circle can be changed.
Since the optimum filling rate lies at around 50%, it can be said that the conveyor system is constructed without a noticeable bottleneck. In addition, the maximum throughput is higher than with the preceding conveyor systems.

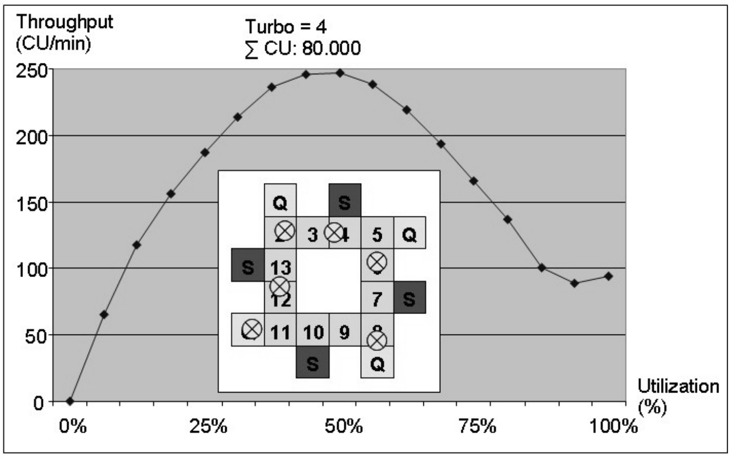


Figure 4.34: CONWIP analysis "circular traffic"

This is based on the fact that several CUs can use different route sections and no point in the system exists through which all CUs must pass.
The distribution of the CONWIP analysis approaches that of a parabola because after a certain injection rate blocking effects occur, which negatively influence the throughput. As the number of CUs in the system increases, the more frequently these interactions occur and hence the throughput drops more sharply. To ensure that the throughput does not drop permanently during injection-rate peaks in an industrial application, the following chapter presents a source module self-regulator that keeps the filling rate at an appropriate level so that the throughput on average stays close to the optimum, even at a high injection rate.

Two intersecting circles

This conveyor system corresponds to the simplest form of two intersecting circles (see Figure 4.35). The variant chosen for this is the one in which a deadlock caused by the control algorithm is impossible because the circles intersect at several modules. For the transport of CUs through this system, no functionalities are required that go beyond those of the circular traffic. However, this conveyor system has a significantly higher level of complexity since several different routes to the sink modules are available.
The analysis shows that the maximum throughput lies at a similar rate as that of the simple circular path. This is a result of the fundamentally similar topologies, whereby several CUs can use a part of the routes for themselves in

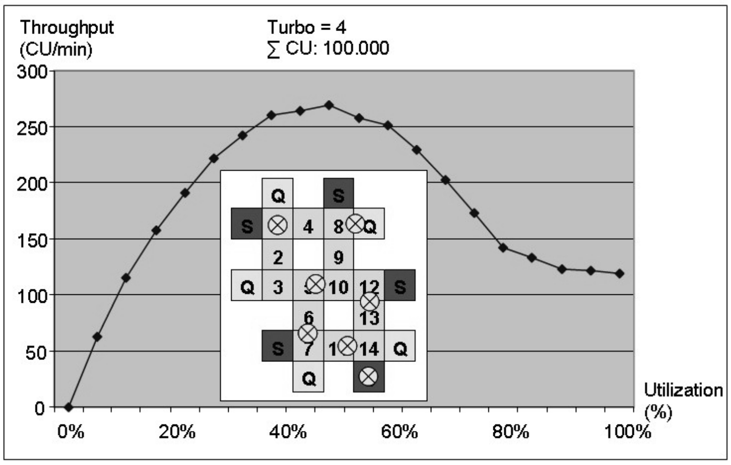


Figure 4.35: CONWIP analysis "two intersecting circles"

parallel, at the same time and independently from each other. The throughput peak at approximately 50% also shows a uniform utilization of all route sections.
The sharp drop in throughput when the optimum filling rate is exceeded demands here again a regulating device in order to limit the dispatch rate.

Complex topology with several circles

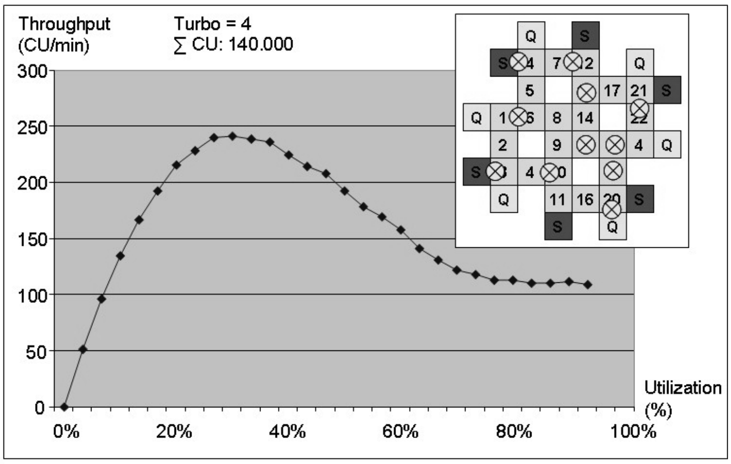


Figure 4.36: CONWIP analysis "complex topology with several circles"

This conveyor system should demonstrate the performance and efficiency of the decentralized control. In spite of a high number of sources, sinks and node points, the CONWIP analysis behaves much like that of the circular traffic. In this case, also, the curve shows a throughput drop, so that an injection regulation would be necessary.

Conclusion of the throughput analysis

The analyses show that the control developed here ensures a high throughput rate for all topology forms. Standstills are successfully prevented by the deadlock check. The throughputs are limited, however, by the minimum distance necessary between the CUs. With more complex conveyor systems, the CONWIP analysis curve approximates a parabola, which is typical when blocking effects start to appear in closed material flow systems, like, for example, a manual picking system with person-to-merchandise.

The analysis indicates that all topologies have an optimum operating point significantly below a filling rate of 100%. Because the control is decentralized, the modules must themselves be capable of reaching the optimum operating point or filling rate, so as to ensure the maximum throughput even when the dispatch rate is too high.

In order to discern the filling rate in a decentralized system, all modules must be continually queried or, at the least, all sources and sinks must be continuously in contact with one another. Subsequently, the CUs in the system must be counted and the current status must be disseminated to the source modules. Finally, these must prevent further injections of CUs into the system when the optimum operating point is exceeded. This procedure is very complex and demands a high level of data traffic in real time.

An analysis performed in the ERP system presents an alternative in which the difference between the number of injected and the number of discharged CUs is calculated. The disadvantage hereby is, on the other hand, that the ERP system, as a central structure, would have to coordinate the injection of units into the system, which would limit the autonomy and flexibility of the system.

Another alternative would be the use of a **source regulator**, which holds back CUs at the sources, in accordance with the answer behavior following transport requests, until the optimum operating point is reached. This procedure is presented in the following chapter.

4.4 Throughput regulation

The system was designed so that the modules are capable of autonomously putting themselves into an optimal operating condition. To achieve this, the system regulates the injection of CUs autonomously. This is done by analyzing the answering behavior of the system when CUs are injected. The module recognizes an overloaded system in that it does not receive an immediate clearance to a request for transport because the neighbor is still occupied. To satisfy the requirement of a uniform programming of all modules, no supplemental functions may be implemented on the source modules. A module instead recognizes its role as a source and sink module on its own as soon as it has only one neighbor.

If a source module doesn't receive a transport clearance several times in a row, the module assumes that the system is overloaded and it increases the waiting time to the next request. This way, the modules have more time to transport the CUs that are already in the system to the sinks before newly-developing vacancies are immediately filled with CUs.

If a source can finally inject a CU into the system, then the time interval is again reduced. Commensurate with the system capacity, over a longer period of time the time interval levels off, so that the system can stabilize itself close to the maximum throughput. The convergence of this regulation strongly depends on the characteristics of the mathematical formula used for the calculation of the individual time intervals between the transport requests (hereafter called the "run-up formula") and the respective layout.

The following three factors play a role for the optimization of the convergence:

- Complexity (e.g. number of circular routes)
- Number and distribution of the sources and sinks
- Transport speed

A run-up formula that improves the throughput for complex as well as for simple conveyor systems must orient itself on two system characteristics. For one, it must be avoided that the source module diagnoses a system overload too early just because other CUs pass by coincidentally at exactly the same time of the transport request. Secondly, when an overload occurs, there should be a fast reaction. The characteristics of the run-up formula were therefore chosen as shown in Figure 4.37.

CUs that are randomly passing by are filtered by sending several requests in short time intervals per transport attempt. The number of requests and the

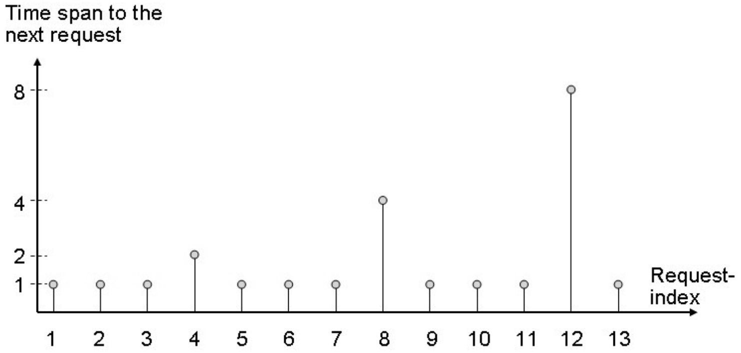


Figure 4.37: Characteristics of the control function of source modules

time intervals between them should orient themselves to the transport speed, so that the total time of the requests t_a is higher than the time required t_m for the passing of a CU. For three requests (as in Figure 4.37),

$$t_{a1} + t_{a2} + t_{a3} > t_m$$

must apply.

In case this request sequence doesn't lead to a clearance, then the module waits for a longer time interval before sending another request. This longer time interval is increased by a given factor after every attempt. In Figure 4.37, the time interval is increased, for example, by a factor of two. The maximum time interval is limited to a certain value, so that the module doesn't wait infinitely before sending a new request. The result of this independent source control is presented below for the "circular traffic" and the "complex topology".

For this analysis, a defined number of CUs per time period are released into the system. The time intervals between the individual CUs are always the same. The source and the corresponding sink for each CU are chosen at random, so that the injection pattern is random. If a source is already occupied with a CU that is waiting to be transported, then it will not be included in the random source selection. If all sources are occupied, then the simulation tool waits until a source is available again. The actual number of CUs injected into the system is reduced accordingly. As the number of CUs is gradually increased, the number of delivered CUs is measured.

As long as the system has a low workload, the number of delivered CUs increases linearly with the number of units entering into the system. If the CUs

begin to interact with one another as more units are injected into the system, the throughput time in fact increases, but these ”local” congestions quickly dissipate again so that the system can process the incoming volume.
If the injection rate is increased more, after a certain time the congestions extend back to the source module. At this point the self-regulating mechanism comes into operation.

Circular traffic

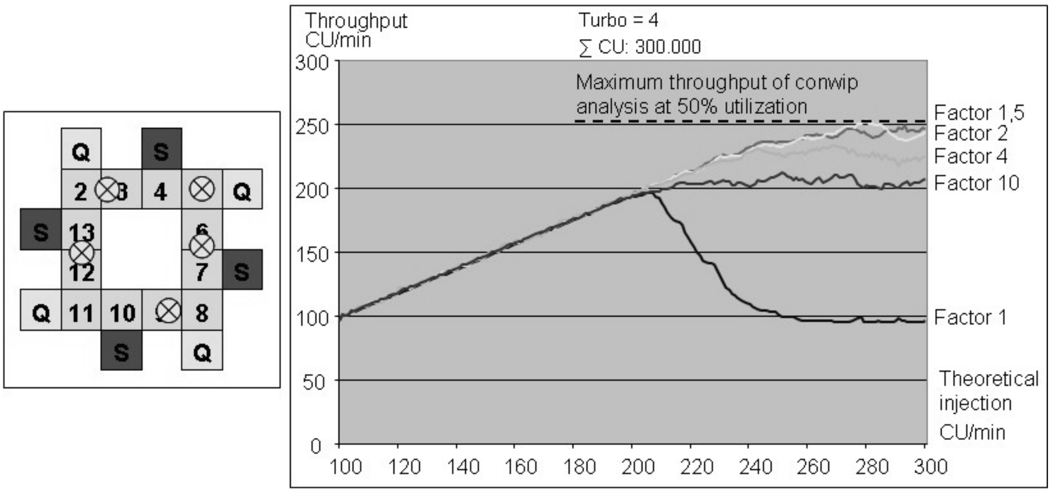


Figure 4.38: Source regulator for ”Circular traffic”

As the injection rate increases, the comparison of the throughput with and without the throughput control mechanism shows a significant improvement even with a wait-time factor as low as 1.5. If the value is increased further, the throughput remains at first at a similar level. However, the dispersion of the individual values increases, because with higher factors, the waiting times become longer so that the system briefly runs dry before a source can again send a new request. In addition, it becomes more difficult for other sources to adjust the waiting times because other source modules potentially get back into a continuous flow again faster and thereby occupy the waiting source’s neighboring modules again with new injections.
When the factor is increased even further, the throughput decreases again because the system runs dry more frequently. The optimum value of the regulator factor depends strongly on the respective layout and an optimum can therefore not be determined for all conveyor systems. During the examination

of many different conveyor systems, a factor of 2, however, proved to be a good compromise.
Furthermore, it was noticed that with a high injection rate and a control factor of 2, the throughput increases up to the maximum throughput of the CONWIP analysis. It could thereby be shown that the source controller assures the highest possible throughput for every injection rate.

Complex topology

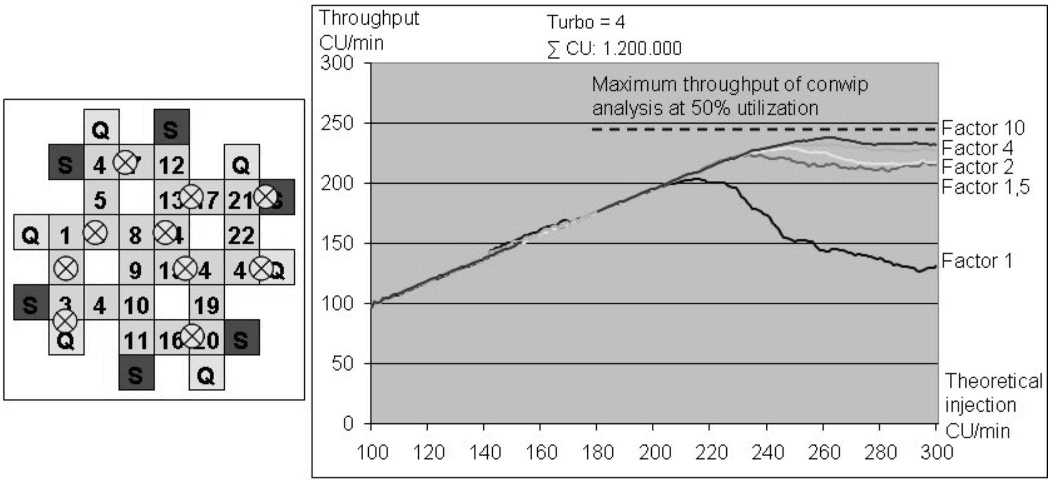
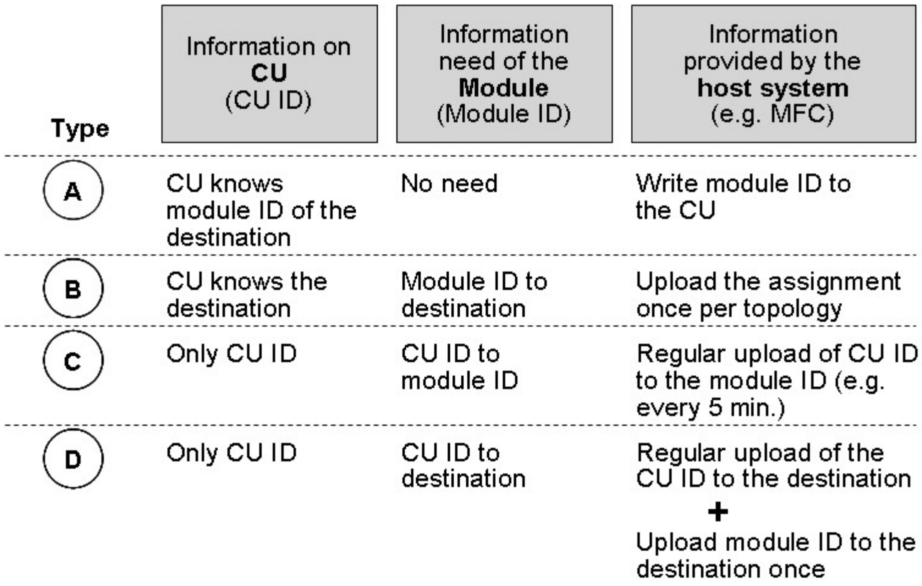


Figure 4.39: Source regulator for "Complex topology"

The analysis of the source regulator for a complex topology produces essentially the same results as the analysis of the circular traffic. However, higher factors should be chosen here because the throughput using a factor of 2 still lies significantly below the maximum achievable throughput. Hence it can be seen that the source regulator becomes even more important as the topology becomes more complex. Without the use of the source regulator, the throughput fall-off with this topology is similarly sharp as with circular traffic.

4.5 Interfaces to the environment

For the use of the decentralized control in an actual industrial environment, it must be assured that the modules have all of the necessary information to move a CU to the sink. Ideally this would mean that the module ID of the respective destination is stored in each CU. This, however, proves to be difficult because the superordinate material flow system must know the topology of the modules as well as have the right to write to the CUs. This, however, contradicts the requirement for a high level of decentralization and autonomy of the system as a whole. To solve this problem, there are several possibilities, which are detailed in Figure 4.40.



Type	Information on **CU** (CU ID)	Information need of the **Module** (Module ID)	Information provided by the **host system** (e.g. MFC)
A	CU knows module ID of the destination	No need	Write module ID to the CU
B	CU knows the destination	Module ID to destination	Upload the assignment once per topology
C	Only CU ID	CU ID to module ID	Regular upload of CU ID to the module ID (e.g. every 5 min.)
D	Only CU ID	CU ID to destination	Regular upload of the CU ID to the destination + Upload module ID to the destination once

Figure 4.40: Information requirements of various types of commissions

Commission type A describes the ideal scenario, in which the module ID of the sink module is stored in the CU. This scenario can be compared to the ”taxi principle”, which is defined as a possible goal for systems in the ”Internet of Things” (Bullinger and ten Hompel 2007). Here the CUs themselves know at which time they must be at which place and inform the respective conveyor means to where they want to be transported. In this case, the CUs must know which module ID corresponds to the desired sink. To achieve this, after the manual construction of the conveyor system, the sink descriptions (e.g., ”Picking area, Station 3”) must be mapped to the module IDs. Before a CU can subsequently be injected into the system, the appropriate module ID according to the mapping table must be communicated to the CU.

The use of this commission type requires a large effort because the regular transfer of the appropriate module ID to every CU is technically complex.

In addition, the high level of flexibility of the decentralized system would be restricted because it would have to be guaranteed that the destination information has arrived at the CU.

Commission type B describes an alternative in which the CU itself knows its destination as above, but it is not necessary for it to have any information about the layout of the conveyor system. To implement this scenario, all that must take place is a mapping of the module IDs to the sink descriptions after the creation of the topology has been completed. This mapping must then be communicated once to all modules. The upload can be done through any module, which disseminates the information autonomously to the other modules in the system.
When a CU is injected into the system during productive operation, the source module assigns the module ID that corresponds to the CU's desired sink and is then able to forward the CU according to the logic used before.
This commission type also places high demands on the independence of the CUs. The big advantage is, however, that a regular information transfer between the modules and the superordinate system does not have to take place. Only status reports from the decentralized systems have to be centrally requested.

Commission C assumes that the CUs have no path-finding information, but rather that they have only an identity (CU ID). The modules need a mapping of the CU ID to the respective destination module's ID. This mapping must be supplied by the surrounding system (e.g., material flow computer). Thereby it is important that the mapping list is transferred on time to the modules before the arrival of the CUs so that they still have sufficient time to forward the mapping to all modules. The one-time mapping of the module IDs to the sink descriptions is stored in the material flow computer and changed as necessary. It is not necessary to store the topology, but simply the module IDs of the sinks.
It would be easiest to implement this type of commission on the material flow computer because the management of the CUs doesn't have to be changed. Ultimately, the material flow computer must simply inform the system in regular intervals to which module the individual CUs are to be transported when they are injected into the system.

Commission type D is similar to commission type *C* with the difference that the mapping of the module IDs to the sink descriptions is stored in the

modules themselves instead of in the material flow computer. The material flow computer must still inform the modules at regular intervals to which sinks the CUs must be transported. However, only the sink descriptions and not the corresponding module IDs are communicated.
This commission type would have the advantage that when the layout is frequently changed, certain modules could be given permanently-assigned sink descriptions and when the topology is changed, the modules would each represent a defined sink. In this way, a module could be permanently assigned, for example, to an assembly team. If the team changes its location in the production area, the team would take its module with it and connect it at another point in the conveyor system. The CUs would immediately be appropriately redirected.

Conclusion of the interface examination
After investigating the different possibilities for the use of conveyor systems with decentralized control within an intralogistic system, it becomes clear that a decentralized approach is also viable here. Supplying the CUs or, respectively, the modules with the necessary information from superordinate control systems does not exceed the previous expenditures. Consequently, the advantages of automatic control can be used without any further disadvantage.

5 Technical implementation

5.1 Introduction of the "Flexconveyor"

In addition to the development of a completely decentralized control, square, structurally identical, continuous conveyor modules were developed during the work on this dissertation. The focus of the first step was on a low-cost integration of all functions within a module. In the second step, several of these modules were built to analyze the system behavior under real conditions. The system is introduced below under the name "Flexconveyor", which stands for "flexible conveyor system".

Basic requirements
For the integration of the mechanical and control functions, a conveyor drive had to be developed according to the specifications detailed in the previous chapters, which can transport conveyor units in four horizontal directions. This function can usually be found in conventional diverters, which, however, mostly just convey in one diversion direction and also do not allow any opposing traffic.
Small load bearers (SLB) are transported as conveyor units, which are allowed an edge length of 200 mm to 400 mm. Therefore SLBs measuring 200 x 300 as well as 300 x 400 can be transported. Furthermore it was stipulated that SLBs cannot exceed a weight of 20 kg and are to be equipped with a re-writable RFID tag located centrally on the base. The underside of the SLB bottoms can be smooth or it can have reinforcing ribs. Therefore the drive has to assure a linear contact to the SLB so that the load can be transported without vibration.

For the identification of the SLBs, the modules must be equipped with an RFID reader, which can read the information on the RFID tag of the SLBs. To ensure reliable communication between the SLB and the module, the reading area must be large enough to read conveyor units that arrive off-centered. On the other hand, the RFID tags of conveyor units that are on neighboring

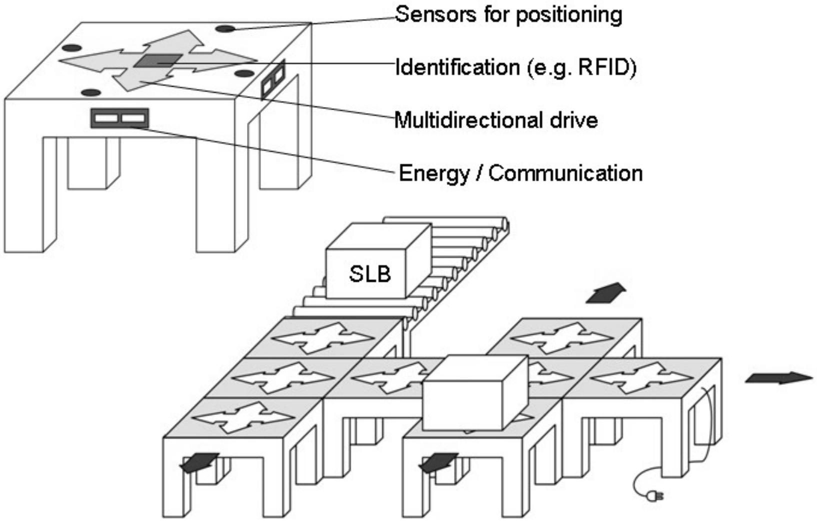


Figure 5.1: Function of individual modules of the Flexconveyor

modules must not be read.
To enable the modules to communicate with each other, the modules must be equipped with four interfaces, one for each transport direction, so that a module can determine from which neighbor an incoming message was received.
To fulfill the basic requirement that only one conveyor unit is allowed on a module at a time, the module borders must be monitored. Photoelectric barriers lend themselves well for this purpose and can monitor each of the four edges. This way the arrival and departure of the conveyor units can be registered. Figure 5.1 shows the basic construction of the modules.

Basic concept of the Flexconveyor
The Flexconveyor module is realized in accordance with the basic requirements as a square module with an edge length of 500 mm. The construction height including all control electronics but without base supports is 120 mm. A mechanical coupling mechanism was intentionally not implemented, so the modules can only be arranged together using lockable wheels attached to base supports.
The drive of the conveyor units along the primary movement axis (preferred direction) is effected by rollers that are arranged at intervals of 100 mm and extend over the entire width of the module. The secondary movement axis (diversion direction) is realized using powered elastic bands, which can be

raised up to the conveyor level using a lifting mechanism.

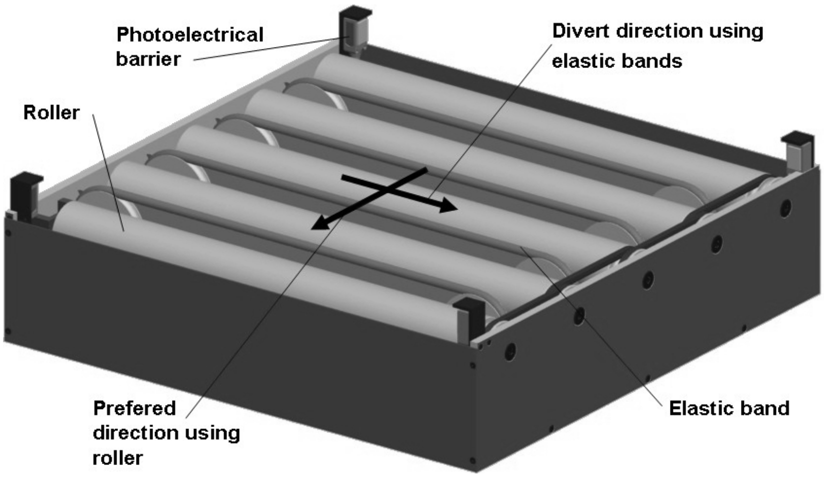

Photoelectrical barrier
Divert direction using elastic bands
Roller
Prefered direction using roller
Elastic band

Figure 5.2: Exterior view of a Flexconveyor module

A specially-developed RFID reader was installed just underneath the rollers. Its antenna encompasses a surface of approximately 200 x 200 mm in the middle of the module. To identify incoming SLBs, a photoelectric barrier was installed on each of the four module edges.
The heart of the Flexconveyor consists of a computer that controls the two motors as well as the photoelectrical barriers. In addition, it has four RS232 interfaces and a memory that can store the control algorithms and the generated routing table. The RFID reader communicates via an RS232 interface with the computer.
The complete module is attached to an adjustable-height base with lockable wheels. These can easily be unlocked to rearrange the layout of the conveyor system within a very short time.

5.2 Construction

5.2.1 Base plate with lifting mechanism

The lifting device of the Flexconveyor is realized using an eccentric, which pushes wedges between the base plate and the support plate of the diverter. With a wedge angle of 30 degrees, the diverter can be lifted 12 mm and due to the leverage of the eccentric the largest part of the power is produced in the upper area of the lifting procedure. The point of transfer between the diversion direction and the preferred direction takes place at the midpoint of the lifting range. Therefore the end positions of the conveyor levels are vertically separated from each other by 6 mm (see Figure 5.3).

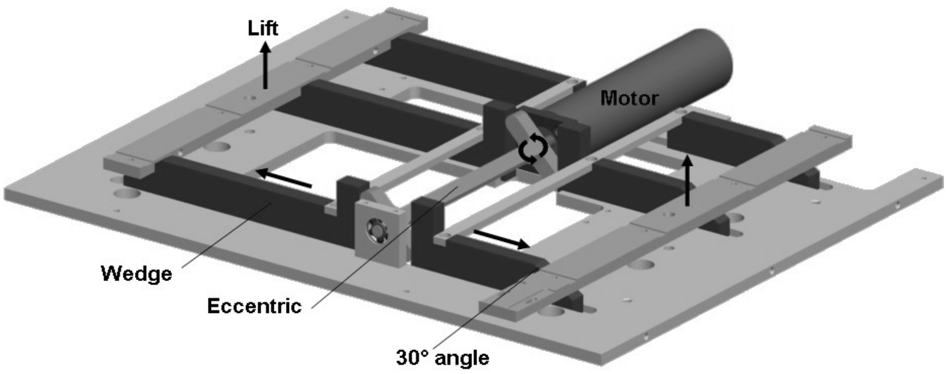


Figure 5.3: Lifting device using the wedge principle

The use of the eccentric allows a high lifting speed and produces the largest power transmission close to top dead center. Moreover, the control effort is kept low because only one reference switch is needed and the motor does not have to reverse.

5.2.2 Diverter with integrated RFID antenna

The support plates of the lifting mechanism each carry four diverter elements, which are able to transport the conveyor unit with elastic bands in the diversion direction. The elastic bands are driven by belt pulleys, which are connected together by a bevel shaft. The bevel shaft is connected through a pair of bevel gears with a friction wheel, which is arranged parallel to the rollers and also carries out the lifting motion (see Figure 5.4).

Through the lifting motion, the friction wheel is pressed against the outer roller, which powers the wheel. Especially because of the eccentric, the most power is produced in the last millimeters of the lift so that the friction wheel pressed very forcefully against the roller. The necessary torque is produced by a roller with an integrated drive motor. When the diverter elements are

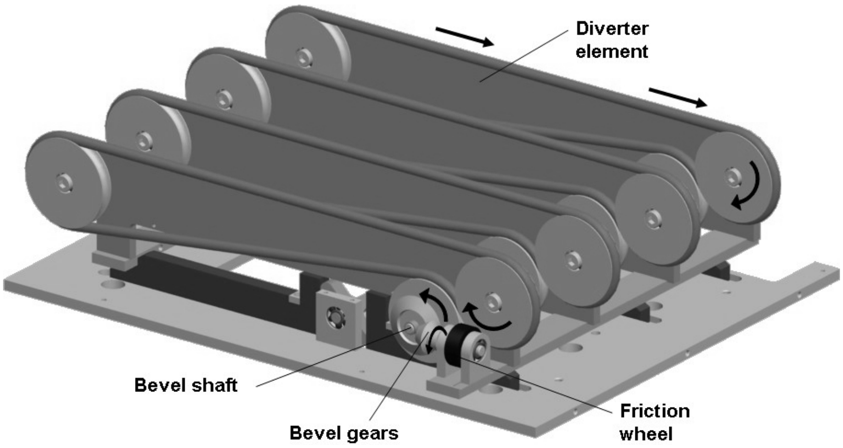


Figure 5.4: Diverter elements, powered by the bevel shaft and friction wheel

lowered, the friction gear does not touch the roller and the driving torque is simply transferred to the parallel-lying rollers (see Figure 5.5).
Through the application of the friction wheel concept, a second motor to power the diversion direction is not necessary, because the roller motor can drive both transport directions.

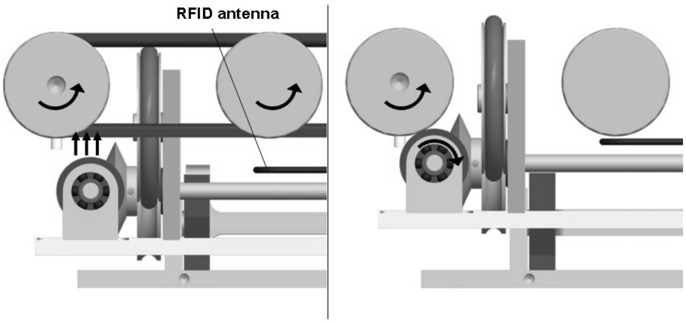


Figure 5.5: Friction connection of the drive roller with the friction wheel to the bevel shaft

The square-shaped RFID antenna is located under the rollers. It is attached through holes in the diverter support. The use of a 13.56 MHz HF system with adequate reading range has the advantage that the electromagnetic interfacing occurs inductively and therefore, despite a metallic environment, the reading range can not grow uncontrollably. This can not be excluded in UHF systems with a larger range. To keep perturbations as low as possible, the diverter elements are completely made of plastic with the exception of the bearings and axles. The positioning of the RFID antenna directly underneath the rollers results in a clearance to the conveyor units of 60 mm (see Figure 5.6).

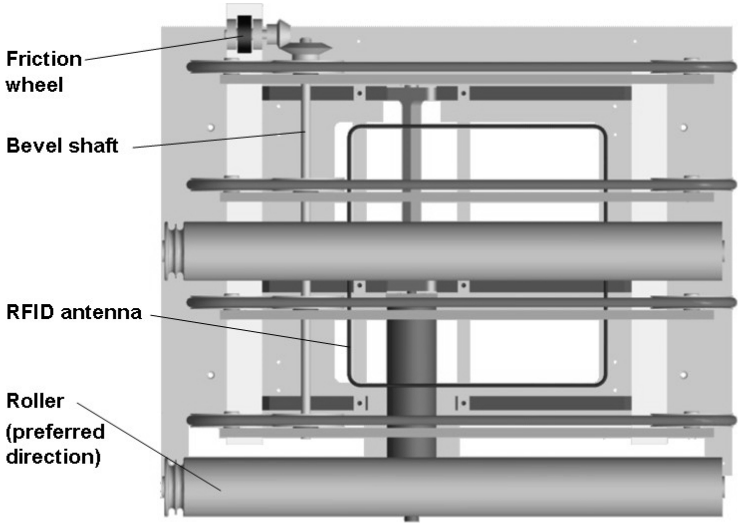


Figure 5.6: Position of the RFID antenna

5.2.3 Roller arrangement and sensor system

The system can transport, without vibration, SLBs with a minimum edge length of 200 mm. For this the SLBs always have to be engaged with a minimum of two rollers. Assuming an even distribution of the weight within the SLB, the minimum distance between two rollers can not be larger than half the edge length (100 mm). For this reason, the Flexconveyor is equipped with five equally-spaced rollers per module, where the middle roller is powered. The passive rollers are driven by so-called poly-V belts, because they are characterized by a low space requirement and high tractive power. The drive motor of the roller produces a speed of approximately $0.4\frac{m}{s}$ with a torque of $0.3Nm$. Commensurate with the ratio of the bevel gears and the friction wheel pair, the speed of the diverter is $0,3\frac{m}{s}$.

To recognize incoming conveyor units, the modules were equipped with four photoelectric barriers, each of which monitors one of the edges from one cor-

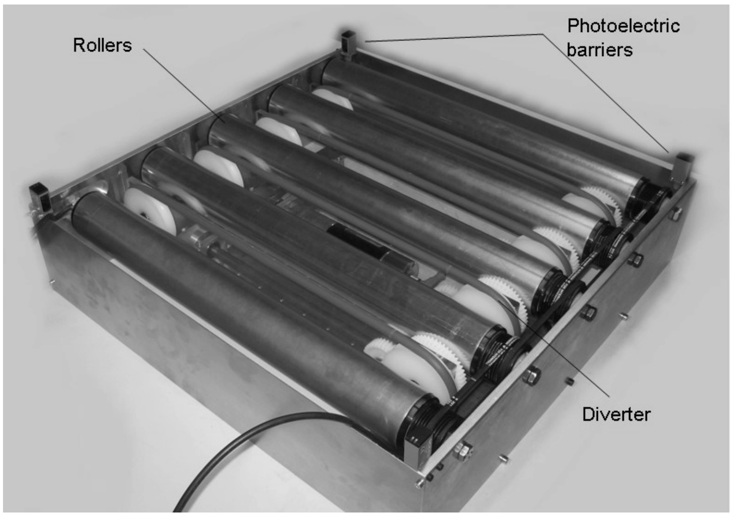


Figure 5.7: Arrangement of the four sensors for monitoring the module edges

ner to the next corner (see Figure 5.7). They also provide a termination report when an outgoing conveyor unit has completely left the module. The photoelectric barriers additionally ensure that a change of direction can only take place when a conveyor unit is completely within the module boundaries.

5.3 Control of the Flexconveyor

5.3.1 Electrical connection

The Flexconveyor is operated with 24V. Because many individual modules are connected together to assemble a conveying system, it makes sense to continuously connect the power supply from module to module. The brief switching-on power draw of the motors of up to 200 W would, with 24 V, result in a current draw of up to 8.3 A. The continuous load is 100 W. Connecting 24 V through several modules would require wire with a very large cross section to avoid a collapse of the voltage when several modules start their motors at the same time. In addition, the loss over long distances would not be economical and a central, sufficiently strong power supply unit would have to be installed. For this reason, each module is equipped with its own power supply unit. This way the supply voltage can be connected through, which results in a significantly smaller wire cross section and lower losses.

Besides, no centrally-installed 24V voltage supply has to be provided.
A microcontroller as well as the RFID reader and a CAN bus interface are connected to the power supply. The USB interface connects all sensors and motors and supplies the equipment with the necessary voltage.
The computer processes the incoming signals from the sensors and processes the control algorithm. Messages are produced thereby that are transmitted through the serial interfaces to the neighboring modules. Incoming messages are also processed accordingly. If an incoming conveyor unit is recognized, the RFID reader is activated and after successfully reading the RFID tags, the corresponding actions are triggered.
Figure 5.8 shows the arrangement of the components and the data protocol used.

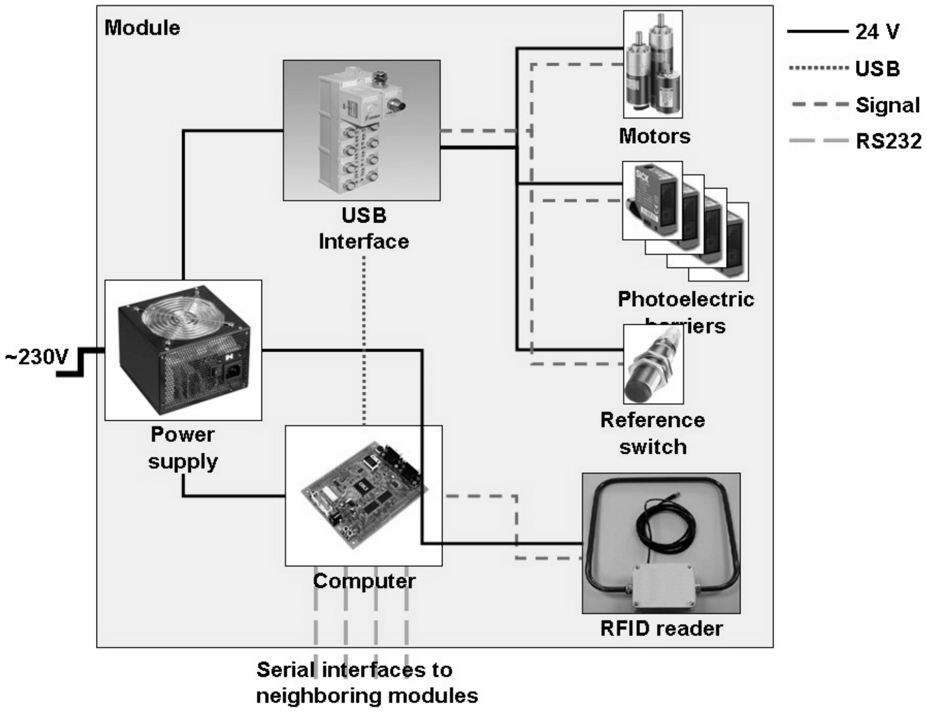


Figure 5.8: Layout of the control components

For an industrial application of the Flexconveyor, the computer will be replaced by a microcontroller board that is expanded so that the sensors and motors can be directly controlled.

5.3.2 Control procedure

The internal operation of the control resembles the operation of a PLC control, which queries the individual input ports sequentially and, when the status changes, triggers actions accordingly. This way changes of status of the sensors as well as incoming messages through the serial interfaces can be processed. The cycle time is less than 5 ms so that even with large systems the reservation runs for new conveyor units entering the system lie within the range of a few seconds.

5.4 Connection of several modules to the topology

Figure 5.9 shows several modules connected together to form a complete conveyor system. At the interfaces to other conveyor systems, appropriate transfer-in and removal points must be installed. Otherwise, the system can operate completely decentralized. To reduce costs, it would be conceivable to remove the diverter units and the two adjacent sensors from non-intersecting modules. The hardware and software should also be identical in this case, because the control logic in straight-ahead sections also contributes to the whole system.

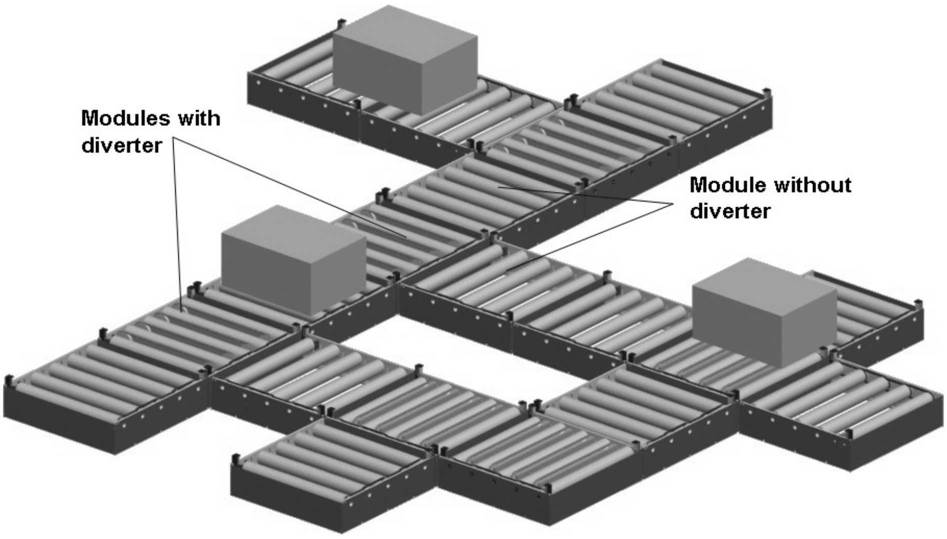


Figure 5.9: Topology example with several Flexconveyor modules

6 Summary

Due to the high investment costs for the installation of conventional conveyor technology in intralogistics, especially smaller companies limit the automation to a minimum. That is why the transportation of unit loads is often carried out by discontinuous conveyors like fork lifts or hand lift trucks. The installation of continuous conveyor systems and the related necessity of a central, individually programmed control are, due to the increasingly shorter product life cycles, increasingly no longer cost effective. A later change or expansion requires high costs and a large time expenditure, whereby a reutilization for other products and processes is economically not possible. The use of driverless transport systems as an automated solution features a higher flexibility, but causes a significantly lower throughput and likewise very high investment and operating costs.

Even though manufacturers of continuous conveyors offer systems that are modularly constructed and are easily connectable, the expenses for an individual adaptation are still high. With conventional, centrally-controlled systems, this individual customizing, for example, requires extensive wiring or, at the least, rewriting the control program. The installation and putting into operation add enormous costs because each system represents a custom system where individual components have to be adapted to the individual requirements.

This dissertation introduces a conveyor system, called "Flexconveyor", that consists of identically-constructed, modules with a decentrallized control, which distinguishes itself from the previous approaches to make the classical automation technology more efficient. The individual modules are autonomous and perform the material flow task collectively. In comparison with rigidly coupled systems, this modular system requires a low planning effort, allows an arbitrary scalability, and a simple exchange of defective modules, whereby the availability of the system is considerably increased and the reusability of the modules is ensured. The capital expenditure is reduced by cost degression with many identically-constructed modules.

The comparison with modern IT networks showed that significant cost savings can be realized with modular, self-controlled systems. Because here, out of a large number of independent nodes, a network is created that can transport large amounts of data without the necessity of an expensive, central computer. Individual computers independently find new connections after being manually connected together and generate in a very short time their own map of the real topology in form of a routing table. When a message is received, the destination is identified and forwarded to the next node on the route. Mechanisms are applied here that ensure that collisions and time losses are avoided.

The comparison of this technology with the requirements of today's material flow systems showed some similarities, which have motivated many conveyor technology research institutions to develop forward-looking material flow systems with decentralized control under the name of "Internet of Things".
A part of the research project with the title "Flexible conveyor systems based on identically-constructed individual modules", which was financed by the German Federation of Industrial Research Associations (AIF), a proprietary procedure was developed on the basis of the knowledge of IT networks to overcome the physical differences between data flow and material flow, and to make possible for the first time the transport of conveyor units, for example, small load bearers (SLB) or paperboard cartons, using a completely decentralized control. In the process, the modules, which are connected together to build the conveyor system, behave very similarly to network routers in IT.

Square conveyor modules, identical in construction, were developed, which each contain an identification system in the form of an RFID reader, a conveyor drive for all four horizontal directions of motion, sensors for the recognition of the position of the conveyor units, and a computer with four serial communication interfaces. All components are networked to and controlled by a mini-computer to avoid expensive components such as a PLC or an industrial PC. The additional expense of equipping each module with an RFID reader and sensors is absorbed by the large number of identically-constructed modules out of which the material flow system is constructed.
The conveyor modules can be manually connected to construct the required topology, whereby the only logical connection between two neighboring modules is a serial RS232 connection.
As soon as the modules are switched on, they send their own module ID to their direct neighbors, who transmit this information together with the num-

ber of modules that have already been registered, on to subsequent modules in turn. This way all modules receive in the shortest amount of time a complete topology overview using the distance vector process, known in network technology.
When a conveyor unit is injected into the system, the destination is identified by the RFID reader. It then searches for the shortest route to the destination in the recently generated routing table and attempts to reserve this route in the layout, whereby it is ensured that the routes are not already reserved by an oncoming conveyor unit. The reservation is done by sending a token, which is sent along the route from module to module. If the token encounters a section already reserved in the opposite direction, it is sent back to the previous module and the routing table is searched for an alternative route.
If a complete route is found, the packet is sent on its way after the following module has cleared the transport. Because multiple packets can be sent in the same direction at the same time, a danger of deadlock exists within a circular path. The system comes to a halt when all modules in the circle already carry a conveyor unit and thereby block each other. This is prevented by having the modules check the route accordingly before starting the transport to the next module.

To investigate the efficiency of systems with a higher complexity and a larger number of modules, a simulation software with the workflow logic of the modules was implemented, which allows freely definable topologies. It was then possible to inject several conveyor units into the system and to observe the material flow. The simulation was used to improve the algorithm and to intentionally cause exceptional situations that the system had to master. Additionally it was proved that when abiding to the basic requirements, a deadlock can never occur.

To avoid a reduction in the throughput due to blocking effects, the system was equipped with a source control mechanism that recognizes the filling rate of the system and, if the load is too high, reduces the injection of further conveyor units into the system. The modules regulate thereby the system filling rate autonomously in order to secure the highest possible throughput. The simulation showed that the throughput is reduced when a certain filling rate is reached because the conveyor units briefly interfere with each other at intersections. The analysis also showed that with high filling rates the material flow never stops completely because deadlocks can be successfully prevented.

The results show that the decentralized control developed here achieves a slightly lower throughput compared with centrally-controlled systems because of the required minimum distance between conveyor units (see the basic requirements, that only one conveyor unit is allowed to be within the module borders at time). Through the modular construction and the decentralized control, however, a heretofore unmatched flexibility in the operation of continuous conveyors was achieved.

In the next stage of development, a larger number of ”Flexconveyor” modules will be constructed to test the system and the already successfully-simulated algorithm under real-life conditions and so to be able to put a completely decentralized and highly flexible material flow system into operation in the near future that is suitable for industrial use.
To improve the control algorithm for industrial applications, a possibility should be developed that allows conveyor units to be sent with different priorities through the system. At least it should be made possible that a maximum throughput time should never be exceeded. This question is the subject of further research that is being conducted at the Institute of Conveying Technology and Logistics (IFL).

References

Arnold, D. (2006). *Intralogistik, Potentiale, Perspektiven, Prognosen.* Springer Verlag Berlin, Heidelberg, New York.

Aurand, A. (2000). *Netzwerkprotokolle in Cisco Netzwerken.* Addison Wesley Verlag.

Bullinger, H.-J. and M. ten Hompel (2007). *Internet der Dinge.* Springer-Verlag Berlin, Heidelberg, New York.

Coffman, E. G. and P. J. Denning (1973). *Operating Systems Theory.* Prentice-Hall, London.

Coffman, E. G., M. Elphick, and A. Shoshani (June 1971). System deadlocks. *Computer Surveys, vol. 3, no. 2*.

Corner, D. E. (2004). *Computernetzwerke und Internets: Mit Internetanwendungen.* Pearson Studium.

Deitel, H. M. (1983). *An Introduction to Operating Systems.* Addision-Wesley.

Elsing, J. (1991). *Grundlagen und Anwendungen der X.200.* IWT Verlag.

Fay, A. and I. Fischer (2004). Dezentrale automatisierungsstrategien für gepächbeförderungssysteme. *Automatisierungstechnik* (7), 335–340.

Fay, A., S. Jerenz, and N. Seitz (2008). Dezentrale steuerung von transportsystemen in analogie zum routing in datennetzen. *Automatisierungstechnik* (6), 284–295.

Furmans, K. and D. Arnold (2006). *Materialfluß in Logistiksystemen.* Springer Verlag Berlin, Heidelberg, New York.

Günthner, W., R. Chisu, and F. Kuzmany (2008). Dezentral und ohne hierarchie. *Fördern und Heben - Report 2008/09*, 6–7.

Günthner, W., R. Kraul, and P. Tenerowicz (2008). Vom prozess zum ereignis - ein neuer denkansatz in der logistik. *Jahrbuch der Logistik*.

Günthner, W. A., M. Heinecker, and M. Wilke (2002). Materialflusssysteme für wandelbare fabrikstrukturen,. *Industrie Management, S.8-10*.

Günthner, W. A. and G. Reinhart (2000). *Abschlussbericht: MATVAR - Materialflusssysteme für variable Fertigungssegmente im dynamischen Produktionsumfeld.* Herbert Utz Verlag, München.

Günthner, W. A. and M. Wilke (2002). Anforderungen an automatisierte materialflusssysteme für wandelbare logistikstrukuren.

Günthner, W. A. and M. Wilke (2003). Materialflusstechnologie - anforderungen und konzepte für wandelbare materialflusssysteme. *21. Dortmunder Gespräche*.

Grötsch, E. (2004). *SPS, Speicherprogrammierbare Steuerungen als Bausteine verteilter Automatisierung.* Oldenbourg Industrieverlag GmbH.

Haaß, W.-D. (1997). *Handbuch der Kommunikationsnetze.* Springer Verlag Berlin, Heidelberg, New York.

Harnisch, C. (2007). *Routing and Switching.* BHV Verlag, Heidelberg.

Hedrick, C. (1988). *Routing Information Protocol.* The Internet Society.

Henshall, J. and S. Shaw (1990). *OSI Explained: End-to-End Computer Communication Standards.* Ellis Horwood.

Jünemann, R. and A. Beyer (1998). *Steuerung von Materialfluss- und Logistiksystemen.* Springer Verlag Berlin, Heidelberg, New York.

Jünemann, R. and T. Schmidt (1999). *Materialflusssysteme: Systemtechnische Grundlagen.* Springer Verlag Berlin, Heidelberg, New York.

Langmann, R. (2003). *Taschenbuch der Automatisierung.* Fachbuchverlag Leipzig im Carl Hanser Verlag München.

Leon-Garcia, A. and I. Widjaja (2006). *Communication Networks - Fundamental Concepts and Key Architectures.* Mc Graw Hill.

Martin, J. (1989). *Local Area Networks.* Prentice Hall.

Messerschmidt, R. and A. Lüder (2002). Industrial ethernet - grundlagen eines standards. *Industrial Ethernet* (B608), 5–13.

Messerschmidt, R. and K. Lorentz (2002). Verteiltes steuerungskonzept für plug-and-play-fähige transportmodule. *A&D Newsletter 99/2002*, 36–39.

Metter, M. (2007). *Industrial Ethernet in der Automatisierungtechnik.* Siemens, Publicis Corporate Publishing.

Moy, J. (1998). *Open Shortest Path First Protocol.* The Internet Society.

Panson, P. A. (1985). *Operating Systems: Structures and Mechanisms.* New York: Academic.

Peterson, J. L. and A. Silberschatz (1983). *Operating system concepts.* Addison-Wesley Pub. Co.

Peterson, L. L. and B. S. Davie (2007). *Computernetze: Eine systemtechnische Einführung.* Dpunkt. Verlag GmbH, Heidelberg.

Pickhardt, R. (2000). *Grundlagen und Anwendungen der Steuerungstechnik.* Vieweg Verlag.

Pigan, R. (2008). *Automatisieren mit PROFINET*. Siemens, Publicis Corporate Publishing.

Pöter, E. and A. Schier (2005). Vitol - vernetzte intelligente objekte in der logistik. *www.iml.fraunhofer.de*.

Rekhter, Y., T. Li, and S. Hares (2006). *A Broader Gateway Protocol 4*. The internet Society.

Rolle, I. (1998). *IEC 61131 - Wozu?* VDE Verlag GmbH.

Saake, G. and K.-U. Sattler (2006). *Algorithmen und Datenstrukturen: Eine Einführung in Java*. Dpunkt. Verlag GmbH, Heidelberg.

Schenk, M. (2002). Fabrikstrukturen mit zukunft. *IRR Deutschland GmbH - Fabrikplanung, Nürthingen, 23.-24. April 2002*.

Schoop, R. (1991). *Beschreibung dezentraler, programmierbarer Steuerungssysteme unterschiedlicher physischer Struktur*. VDI-Verlag.

Schürmann, B. (2004). *Grundlagen der Rechnerkommunikation: Technische Realisierung von Bussystemen und Rechnernetzen*. Vieweg + Teubner Verlag.

Schroer, W. (2005). Multishuttle - universell einsetzbar. *Hebezeuge und Fördermittel* (1-2), 34–35.

Seemüller, S. (2006). *Durchsatzberechnung automatischer Kleinteilelager im Umfeld des elektronischen Handels*. Herbert Utz Verlag, München.

Tanenbaum, A. S. (2002). *Moderne Betriebssysteme*. Pearson Studium.

ten Hompel, M. (2007). Zellulare fördertechnik. *Logistik entdecken*, 6–9.

ten Hompel, M. and M. Corban (2004). Nicht alle agenten gehören zu scotland yard. *Industrieanzeiger* (46), 42–43.

ten Hompel, M., S. Libert, and U. Sondhof (2006). Dezentrale streuerung für materialflusssysteme am beispiel von stückgutförder- und sortieranlagen. *Logistics Journal*, 1–9.

ten Hompel, M. and L. Nagel (2008). Zellulare transportsysteme - den dingen beine machen im "'internet der dinge"'. *it Information Technology* (1), 59–65.

ten Hompel, M., A. Schier, and E. Pöter (2008). Dezentrale materialflusssteuerung unter einsatz von drahtlosen sensornetzwerken. *4. Fachkolloquium der Wissenschaftlichen Gesellschaft für Technische Logistik e.V.*.

ten Hompel, M., P. Stuer, and D. Liekenbrock (2004). Echtzeitnahe steuerung von materialflusssystemen auf basis autonomer agenten und entitäten. *Realtime Logistics*.

van Brussel, H. and P. Valckenaers (2000). Holonic manufacturing systems and mulli agent manufacturing control. *Pro: 9th IMCC, Hong Kong*.

Warnecke, H.-J. (1995). *Aufbruch zum fraktalen Unternehmen: Praxisbeispiele für neues Denken und Handeln.* Springer Verlag Berlin, Heidelberg, New York.

Wellenreuther, G. and D. Zastrow (1998). *Steuerungstechnik mit SPS.* Vieweg + Teubner Verlag.

Wellenreuther, G. and D. Zastrow (2008). *Automatisieren mit SPS - Theorie und Praxis.* Vieweg + Teubner Verlag.

Westkämper, E., H.-H. Wiendahl, and P. Balve (1998). Dezentralisierung und autonomie in der produktion. eine systematische betrachtung der klassifizierungsmerkmale. *ZWF 93* (9), 407–410.

Wiendahl, H. P. (2005). Hanser Verlag, München.

Wildemann, H. (1988). *Kundennahe Produktion durch Fertigungssegmentierung.* Hanser, München.

Wilke, M. (2006). *Wandelbare automatisierte Materialflusssysteme für dynamische Produktionsstrukturen.*

Wilke, M. (2008). Dezentral steuern, zentral kommunizieren - ein steuerungskonzept für wandelbare materialflusssysteme. *Logistics Journal*.

List of Figures

Universität Karlsruhe (TH)

Institut für Fördertechnik
und Logistiksysteme (IFL)

IFL

To increase the flexibility of continuous conveyor systems, a completely decentralized control system for a modular conveyor system was developed. The system is able to carry conveyor units without any centralized infrastructure. Based on existing methods of data transfer in IT networks, single modules operate autonomously and, after being positioned into the required topology, independently connect together to become a functioning conveyor system. Parallel to the development of the decentralized control system, identical square modules were designed. To fulfill the task, every module was equipped with an RFID identification system, sensors, a multidirectional drive, and a microcontroller-based control unit that executes the control algorithm.

The following functions can be performed by these modules:

- Independent generation of the topological map in the form of routing tables
- Recognition of an incoming conveyor unit and identification of the destination address
- Planning of the path to the destination taking into consideration conveyor units already located in the system
- Protection against collisions and deadlocks, and transportation of the conveyor unit to the next module
- Autonomous regulation of the injection rate to ensure the highest possible troughput

The throughput performance of the control algorithm developed here was analyzed by simulating representative topologies. Furthermore, it was proven that under certain conditions, despite the conveyor routes being used in multiple directions, a situation can never arise where conveyor units block each other and the flow of material comes to a halt in the form of a deadlock.

ISSN: 0171-2772
ISBN: 978-3-86644-366-2

www.uvka.de