An Analysis of Real-Time Ethernets With Regard to Their Automatic Configuration

Lars Dürkop[1], Jürgen Jasperneite[1,2] and Alexander Fay[3]

[1]inIT – Institute Industrial IT, Ostwestfalen-Lippe University of Applied Sciences, D-32657 Lemgo, Germany {lars.duerkop, juergen.jasperneite}@hs-owl.de
[2]Fraunhofer IOSB-INA Application Center Industrial Automation, D-32657 Lemgo, Germany juergen.jasperneite@iosb-ina.fraunhofer.de
[3]Institute for Automation Technology, Helmut Schmidt University, D-22043 Hamburg, Germany alexander.fay@hsu-hh.de

***Abstract*—Future market conditions require production systems which can be easily adapted to changing demands. However, the engineering process of industrial automation systems is characterized by high manual configuration efforts. Thus, the reconfiguration of such system leads to time-intensive and expensive downtimes. Therefore, this paper will present a concept on reducing the engineering effort – at least on the lower layers of the automation pyramid. Due to the real-time requirements on these layers, specific communication technologies must be used there – for example Real-Time Ethernets (RTEs) which are increasingly applied in industrial automation. However, their real-time capability is contrasted by an additional configuration effort in comparison to standard networks from the information technology domain. This paper will show an approach for the automatic configuration of RTEs and will check its applicability on the most widely-used RTE variants.**

I. Introduction

The manufacturing industry is facing increasing competitive pressure due to rapidly changing customer demands, short product development cycles or fast changing technologies. So the dynamic adaption of industrial production systems will be one of the industry's key challenges in the future [1]. Mehrabi formulates the new demands as follows [2]: "The manufacturing systems [...] must be rapidly designed, able to convert quickly to the production of new models, able to adjust capacity quickly, and able to integrate technology and to produce an increased variety of products in unpredictable quantities."

Currently, the engineering process for automation systems is characterized by high time-consuming manual configuration efforts. For example, all devices present in the automation network, their properties and the data to be transferred (including the timing of the data) must be defined by an automation engineer. Any modification of such a system requires an at least partial manual reconfiguration. Therefore, the automation engineering is a major obstacle on the way to future production systems. Indeed, the development of new or modified engineering approaches reducing the configuration effort is a highly relevant research topic [3].

By today, the concrete engineering steps differ according to the respective hierarchy level of the automation system. An overview of the different levels is given by the automation pyramid in figure 1. At the field level sensors and actuators are forming the interface to the controlled physical process. At the control level Programmable Logic Controllers (PLCs) read values from the sensors, process them and generate new signals for the actuators. This process is repeated cyclically and autonomously on the basis of a pre-defined control logic. At the operations level the data of a plant is collected and visualized, typically by Supervisory Control and Data Acquisition (SCADA) systems. Finally, the enterprise level performs the business administration like production planning or material ordering. At this level standard computers and IT technologies are used. From the communication aspect, the levels differ in their need for real-time communication. This has also impact on the engineering process: The higher the real-time requirements, the more important the communication engineering is.

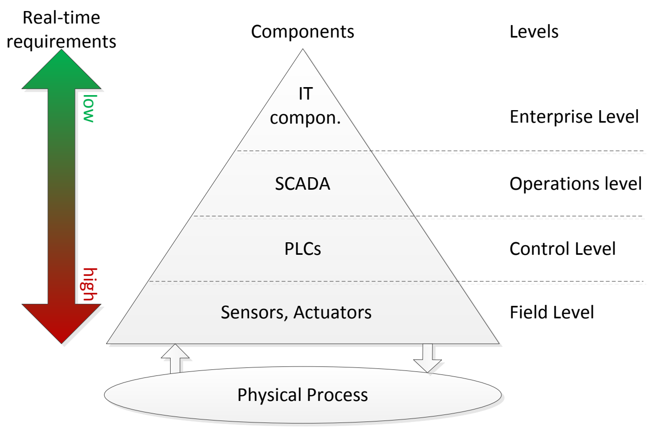

Fig. 1. Automation pyramid

The focus of this paper is on reducing the engineering effort needed in the two lower levels – and here especially on reducing the communication engineering. Whereas on the operations and enterprise level standard IT networks and protocols like Ethernet and TCP/IP can be used, the transmission path between sensor, PLC and actuator is part of a time-critical control loop. Depending on the nature of the controlled physical process, the communication technology must fulfill high real-time requirements. In general, three real-time classes can be distinguished in Ethernet-based industrial communication [4]:

- Class 1: Data traffic on top of TCP or UDP, cycle times of about 100 ms, used for engineering and for process monitoring.
- Class 2: Data traffic on top of Ethernet's media access control (MAC) layer, cycle times 1 – 10 ms, used for process control.
- Class 3: Synchronized data traffic on top of a modified MAC layer, cycle times 250 µs – 1 ms, jitter less than 1 µs, used for motion control.

The requirements of classes 2 and 3 cannot be fulfilled by Ethernet in conjunction with TCP/IP [5]. Therefore various real-time adoptions to the Ethernet standard subsumed under the term Real-Time Ethernet (RTE) have been introduced. The fieldbuses of the first generation are currently replaced or supplemented by RTE. The different RTE variants have in common that the TCP/IP protocol family can be used in parallel to the real-time data traffic. This opens up the possibility of using the same network infrastructure from the office and IT level down to the field level. However, as described in section III-B, a high manual configuration effort is needed to bring RTEs into a communication-ready state. This is in contrast to IP-based communication in standard Ethernet where the only prerequisite for an automatic start of communication is a protocol for address assignment like the Dynamic Host Configuration Protocol (DHCP). For this reason the engineering effort for the integration of field devices into the process control level comprises not only application-related aspects, but also a significant level of communication engineering.

This point leads to the main focus of this paper: The development of methodologies for the automatic configuration of Real-Time Ethernets. This topic represents a partial aspect of the overall objective of reducing the engineering effort. This paper is structured as follows: In section II related work is summed up. An introduction to RTEs and their autoconfiguration is given in sections III and IV. An analysis of the most common RTE variants with respect to their suitability for the autoconfiguration approach is carried out in section V. Based on the results of the analysis, an autoconfiguration procedure is presented in section VI. The paper ends with a conclusion in section VII.

II. Background and related work

The reduction of engineering effort in the domain of industrial automation is a key aspect in many research projects. In general, one can distinguish between approaches introducing a completely new automation architecture and approaches adapting existing structures.

An example of the first group is the proposal to use Service-Oriented Architectures (SOAs) in the automation domain [6]. SOAs –a software design pattern originating from the IT world– are based on services, each representing a certain functionality. Applications in a SOA are built as a composition of different services whereby each service exposes only its functionality to other services, the implementation is not visible from outside the service. Furthermore, all services are loosely coupled: They operate independently from each other, their interactions are stateless, asynchronous and not context-related [7].

In the context of industrial automation the services represent the functionalities of individual field devices or production modules. The behavior of the overall system is controlled by the coordination of all services. Due to the loose coupling of SOA-based automation components the reconfiguration effort for modifying existing automation structures shall be minimized – whilst in current systems all components are linked to each other in a static manner which obstructs changes [8]. Even the initial setup of automation systems shall be simplified by the use of SOAs [9].

On the communication layer almost all SOA solutions in the automation domain either use the Devices Profile for Web Services (DPWS) or the OPC Unified Architecture (OPC UA). Comparative descriptions of both protocols can be found in [10] and [11].

Beyond the communication aspect, the services must be orchestrated to compose an automation process. Therefore several approaches can be found in the literature: In the European SOCRADES project and in its successor IMC-AESOP formal methods and tools for the service orchestration using High-Level Petri Nets have been developed [12]. In [13] constraint satisfaction problem solvers are used to generate production plans based on configuration and maintenance data provided by services. However, these methods only support the orchestration process – the actual composition must still be performed manually. Approaches for the automated orchestration are using semantic annotations of services. Based on an abstract process description, an orchestration engine searches and connects the services whose semantic descriptions are in compliance with the requirements of the abstract definition [14].

Apart from the mentioned advantages, the SOA approach lacks the support of real-time communication – especially of the real-time classes 2 and 3. This is partly due to the fact that the technologies used for the implementation of SOAs do not target suitable real-time constraints [15]. Furthermore, it is questionable whether the central SOA principle of loose coupling is applicable to the field level of industrial automation systems [7]. An alternative approach is the integration of well established solutions for real-time communication like RTE into SOAs. For example, the RI-MACS [15] and the iLAND [16] projects suggest to extend a SOA by a dedicated real-time channel. However, further implementation details are not provided.

As mentioned, the main concern in using RTEs is the necessary manual configuration – which is contrary to the intended objective of reducing the engineering effort. Therefore, methods for the automatic configuration of RTE are required. Their field of application will not only be limited to SOA environments – instead, RTE autoconfiguration mechanisms can be the foundation for any attempts in simplifying or automating the setup procedure in current automation systems.

There are already several attempts for an automatic configuration of RTEs. In [17] a procedure for the RTE Powerlink is presented exploiting Powerlink-internal functions. A similar approach is used for the autonconfiguration of Profinet IO in

[18]. In addition, that paper introduces the concept of an ”ad-hoc channel” used for configuration purposes based on the non-real-time channel of the RTE Profinet IO. It utilizes the SOA technology DPWS for discovering devices and ascertaining their capabilities. In [19] DPWS is replaced by OPC UA and a mechanism for the autoconfiguration of modular Profinet IO devices is presented. A similar concept is shown in [20].

However, the mentioned publications do not consider the transferability or generalizability of their approaches. For example, the ad-hoc channel concept is realized within the RTE Profinet IO – it has not been checked whether it can be adapted to other RTEs. Therefore, this paper will derive general functionalities needed for RTE autoconfiguration. A comparison of the most common RTE variants will check how these functionalities can be realized within the individual RTEs. Thereby, RTE-independent mechanisms should be used so far as possible.

III. Fundamentals of Real-Time Ethernet

Real-time-capable fieldbus technologies were introduced in the industrial automation about two decades ago. In the meantime Ethernet and the TCP/IP protocol stack, which both are non real-time-capable, have become the dominant standards for local area networks in the home and office environment. Fieldbuses and Ethernet are not compatible and so field devices could not be accessed from networks outside the automation world. To overcome this barrier modified Ethernet-based technologies are currently finding their way into the industrial automation. The different real-time modifications, which are not compatible to each other, are referred to as Real-Time Ethernet (RTE).

A. Categories of Real-Time Ethernets

The key differentiator between all RTEs is the supported real-time class. According to the defined classes in section I RTEs can be divided into three categories which differ with regard to their compatibility with Ethernet and TCP/IP (see figure 2):

- Category A: Used for real-time class 1. Both non-real-time (NRT) and real-time (RT) applications are based on the TCP/IP stack and standard Ethernet. Example: Modbus TCP.
- Category B: Used for real-time class 2. The TCP/IP stack is omitted for RT data. This leads to significant improvements of transmission times since studies have shown that the major part of the end-to-end delay occurs within the TCP/IP stack of network nodes [21]. On the hardware side, standard Ethernet devices can still be used. Example: Profinet IO Class B.
- Category C: Used for real-time class 3. The TCP/IP stack is omitted for RT data. Modified Ethernet components (i.e. controller, switches) must be used. Example: Profinet IO Class C, Ethercat, Ethernet Powerlink.

On the field level, only the categories B and C are relevant. Category A RTEs will thus not be considered in the following.

Besides the supported real-time classes, RTEs differ in several other aspects like throughput, media access, topology, time

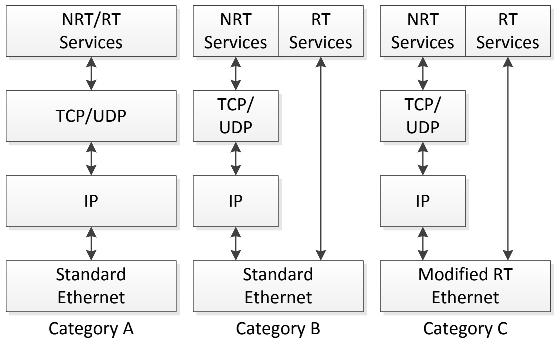

Fig. 2. RTE categories [22]

synchronization, addressing and the communication model. A comprehensive overview can be found in [23]. According to the issue of this paper, the focus of the descriptive RTE comparison in section V will be on the RTE characteristics which are relevant for the automatic configuration approach.

In order to be able to identify the relevant properties, the general requirements on an automatic RTE configuration will be derived in the next sections.

B. Engineering of RTE-based networks

RTEs are, as other industrial communications systems like fieldbuses, used for process data exchange between an application on the control level (typically process control software executed by a PLC) and devices on the field level. Accordingly, the communication structure of all considered RTEs consists of one central node (called controller or master) coordinating the data transfer to all other nodes (called IO-devices or slaves). Usually the PLC performs the role of the controller and the field devices represent the slaves. The engineering flow currently needed for setting up RTEs is depicted in figure 3.

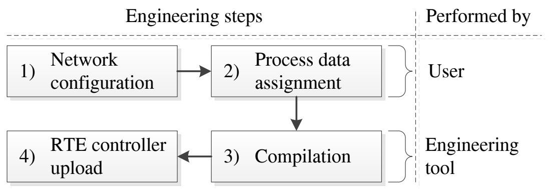

Fig. 3. Engineering flow for RTE setup

The RTE commissioning process is assisted by an engineering tool usually coming from the PLC manufacturer. The automation engineer (in its role as the tool’s user) has the task of providing all necessary information to the tool. Afterwards it calculates the parameters needed by the RTE controller. The individual steps are:

1) **Network configuration**. This step includes the identification of the installed RTE devices. Therefore, the engineer must add RTE-specific device description files (DDFs) of the devices present in the network to the engineering tool. Furthermore he has to perform, among other configuration activities, the address assignment as well as the specification of data cycle times.

2) **Process data assignment**. Here, the engineer links the communications objects of the control application to the according objects of the field devices. For instance, on the application side these objects can be represented by variables and on the field device side by input/output-registers.
3) **Compilation**. After gathering all information about the network structure, the devices and the data to transfer the engineering tool generates the detailed communication parameters. The outcome of the compilation are, for example, frame structures or a communication schedule. The tool-internal compilation process is part of the vendor's intellectual property and is not disclosed to third parties.
4) **RTE controller upload**. The compiled configuration file from the previous step is uploaded to the controller of the RTE – which in turn configures the other RTE devices by using the information provided in the configuration file. The file format and the interface between engineering tool and controller are usually proprietary.

These four steps allow the derivation of the requirements on an automatic RTE configuration as it follows in the next section.

IV. BASICS ON RTE AUTOCONFIGURATION

As mentioned in section III-B, four configurations steps are necessary before real-time communication over an RTE can be realized. The aim of the autoconfiguration approach is to supersede these steps. At the end, the communication engineering shall be performed completely by the autoconfiguration mechanism. This approach is similar to the "Plug & Play" functionality of the Universal Serial Bus (USB): When an USB device is getting connected to a host, the bus will be configured automatically and standardized communication channels between host and device are established.

To reach a similar behavior in the case of RTE, the manual RTE configuration must be automated. Therefore, on the one hand, the autoconfiguration mechanism must clone the functionality of the RTE engineering tool (see steps 3 and 4 of figure 3). On the other hand, the information traditionally provided by the engineer (see steps 1 and 2 of figure 3) must be gathered automatically. One possible architecture allowing to obtain this information is shown in figure 4 [7].

Here, the information normally provided by the engineer is collected by the **discovery** function block. According to figure 3, this information comprises the network configuration and the process data assignment. The required network configuration data (i.e. which devices are installed?) is RTE-dependent. Which information is necessary in detail and how it can be obtained for different RTEs is analyzed in section V.

For the automatic process data assignment, which is independent from the used RTE variant, the discovery block must determine which variables belong to which field devices. Currently, there are rare approaches for this problem. One idea is based on the semantic **self descriptions** of control application and field devices [7]. In that method the self description of each field device contains information about the device's type. For example, a device measuring the temperature

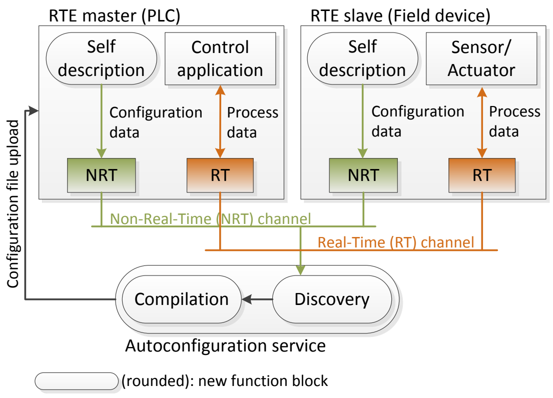

Fig. 4. Autoconfiguration system architecture

can identify itself as a "temperature type sensor". On the other hand, all external variables of the control application must be annotated by using according types – in this example the variable would be described as a "temperature type variable". Then, the discovery block compares the self descriptions of the control application and the field devices and connects the temperature variable to the temperature sensor. It should be noted that this approach needs a common type ontology used for the description of the control application as well as of the field devices. Furthermore, ambiguities (ie., two temperature-type variables must be connected to two temperature sensors) have to be resolved manually, yet.

To retrieve the semantic self descriptions the discovery block must communicate with each device – although the RTE is not configured yet at this stage and, therefore, at least real-time communication is not available. Since the self descriptions need not to be transferred in real-time, the use of the non-real-time (NRT) channel available in all RTEs seems obvious. However, not all RTEs offer their NRT capability before they have been configured. The RTE comparison in section V will check under which conditions NRT traffic can be used in the different RTEs.

For the technical realization of the self descriptions SOA functionalities can be exploited as they include methods for device description. For example, the SOA implementation OPC UA offers an object oriented approach for describing data in an information model. OPC UA also includes a communication protocol for accessing the model. Another SOA implementation for the industrial domain, DPWS, offers similar possibilities. The concrete procedure used in both implementations for the retrieval of device descriptions can be found in [11]. As almost all SOA implementations, both OPC UA and DPWS are based on TCP/IP. Therefore, the prerequisite for both protocols is an operational NRT channel for TCP/IP communication.

The **compilation** function block makes use of the information collected by the discovery block. It has to provide the functionality normally provided by the engineering tool (see *Step 3 - Compilation* in section III-B). Due to the complexity this paper will not provide details on the compilation process of each RTE. Further information, for example on the Profinet IO Class B compilation process, can be found in [19]. The computation of a communication schedule for Profinet IO

Class C is explained in [24]. Instead, this paper is focused on the analysis of which information has to be available as a precondition for the compilation process.

The following section will investigate how the autoconfiguration process can be realized in different RTE variants. Note, that –independent from the used RTE– each field device could require additional parametrization, for example sensor calibration data has to be provided. The automatic generation of this data is not included in the provided autoconfiguration approach.

V. Applicability of the autoconfiguration approach on concrete RTEs

The last section has given a general introduction to the mechanisms needed for RTE autoconfiguration. On this basis, this section will analyze different RTEs variants with regard to a concrete implementation of the autoconfiguration approach. The important issues in this context are:

1) NRT channel: What steps are necessary to enable non-real-time TCP/IP communication used for the retrieval of the self descriptions?
2) Discovery: Which network configuration data is necessary for the compilation process and how can it discovered automatically?
3) Hot-plugging: When the RTE is in its operational state, is it possible to add new devices without interrupting the real-time communication?

In the following, these questions are addressed for the five most common category B and C RTEs in the industrial automation domain which are Profinet RT, Profinet IRT, Ethernet/IP, Powerlink and Ethercat [25].

The first and the third question will be answered by analyzing the RTE protocol. Therefore a short introduction to the respective RTE is given, whereby the focus is on the media access method since the prerequisites for the NRT channel as well as the hot-plugging capability are closely linked to the media access. In the context of the second question (discovery of network configuration data), the examination of the RTE startup phase could be helpful. In each RTE, the controller send startup frames to its slaves. These frames include all parameters necessary for RTE commissioning. However, it is not always possible to determine if the parameters have been calculated by the engineering tool (in this case these parameters can be set automatically by the autoconfiguration service, too) or a user-input was necessary for the individual parameter. Therefore, a practically approach has been chosen: The engineering tools available for the different RTEs have been checked to see what parameters can be set by the user. For reasons of simplification only parameters have been considered which must be set, optional or parameters with a default value have been omitted.

A. *Profinet IO Class B*

Profinet IO Class B –in the following: Profinet RT– is a category B RTE supporting real-time class 2. It is designed for cyclic data transfer between one IO-Controller and several IO-Devices. The latter can be designed modular, whereby the user can chose the sub-components of such a device.

The following three paragraphs are related to the three questions mentioned above.

1) Media access & NRT channel: Profinet RT and standard Ethernet devices can be used in the same physical network as both use the standard Ethernet media access mechanism. Furthermore, Profinet RT devices can be accessed by TCP/IP at any time and, therefore, TCP/IP communication can be used without prior parametrization of the Profinet network. However, DHCP usually used in IP networks for address assignment is not part of the Profinet standard and is not supported by most Profinet devices. Instead, the autoconfiguration service must implement an own dynamic address allocation method for ensuring IP connectivity. Therefore, the Profinet Discovery and Configuration Protocol (DCP) can be used. DCP offers possibilities for device discovery and for address assignment. The real-time capability of Profinet RT is achieved by using a priority tag in the Ethernet header, whereby Profinet RT frames are prioritized in the switches of the network.

2) Required information: In the conventional approach for setting up a Profinet RT network, the user must provide the installed devices, their device names and their IP addresses. In the autoconfiguration scenario, this information must be gathered automatically. During the establishment of the NRT channel DCP has been used for dynamic device discovery and address allocation. So, the installed devices and their addresses are known. Furthermore, the DDFs of each device are needed by the autoconfiguration service. These files, in Profinet called Generic Station Description (GSD) file, can be obtained by accessing a central GSD database, for example. The correct file for a device can be identified by using vendor and product IDs. The IDs are stored inside each GSD file and can be requested from the devices by DCP during the address assignment process. In the case of modular devices, it is further necessary that the individual modules attached to a device are known. These can be identified by using the Profinet command *Read Implicit Request* as described in [19].

3) Hot plugging: Hot plugging can be realized by cyclically broadcasting DCP identify requests. The autoconfiguration service can detect newly attached devices by their DCP identify response. After assigning an IP address by DCP, IP connectivity is given and the self configuration of the new device can be obtained by using the mentioned SOA protocols OPC UA or DPWS.

B. *Profinet IO Class C*

Profinet IO Class C –in the following: Profinet Isochronous Real-Time (IRT)– is a category C RTE which supports real-time class 3. It differs from Profinet RT in its strictly deterministic data traffic. This results in a more complex engineering process.

1) Media access & NRT channel: Profinet IRT divides the communication into three phases: In the isochronous phase the communication is exactly scheduled: Each device sends its data at pre-defined times. Therefore all devices must be synchronized in time. The isochronous phase is followed by a phase for Profinet RT and, at last, by a phase for non-real-time data. Before the isochronous communication has been configured by the IO-Controller, a Profinet IRT network behaves like a Profinet RT network. Accordingly, the NRT channel can be realized as in Profinet RT.

2) Required information: In addition to the data required in Profinet RT, in the IRT variant the network topology must be known for the calculation of the communication schedule. The autoconfiguration service can discover the topology by using the Link Layer Discovery Protocol (LLDP) which is mandatory in Profinet IRT. The communication planning can be based on the algorithm published in [24].

3) Hot plugging: New devices can be detected by DCP identify requests as described in the Profinet RT section as long as the network topology between already existing devices is not changed. A new device can be attached to an unused switch port, for example. The switch buffers incoming non-IRT packets and enqueues them into the non-real-time phase. However, after detecting a new device the communication must be re-scheduled which results in an interruption of the isochronous data transfer.

C. Ethernet/IP

Ethernet/IP (EIP) uses the TCP/IP-based Common Industrial Protocol (CIP). In combination with the additional CIP Sync, which introduces hardware-based time synchronization, Ethernet/IP is a category C RTE supporting real-time class 3.

1) Media access & NRT channel: EIP combines properties of Profinet RT (prioritization) and Profinet IRT (synchronization). Implicit Messages, used for transferring cyclic process data, are prioritized by an Ethernet priority tag. In addition, they contain a timestamp containing the acquisition time for input data. For output data, the timestamp specifies the time of data execution. However, in contrast to Profinet IRT, is not ensured that the packets receive their recipient on time. The Precision Time Protocol used for synchronization allows the conflict-free use of Ethernet/IP and standard Ethernet devices in the same network. Thus, the NRT channel can be realized without any additional effort. The address assignment can be carried out by using usual IP-based methods like DHCP.

2) Required information: Like in Profinet, in EIP the installed devices and their addresses must be known. Since IP connectivity is available by default, this information can be obtained by using SOA-based discovery methods (see phase 2 in section VI). Furthermore, also in EIP the DDFs, here called Electronic Data Sheet (EDS), are necessary. The appropriate files can be identified by their Vendor ID, Device Type and Product Code. The autoconfiguration service can obtain these parameters from the EIP devices by using the *ListIdentify* command.

3) Hot plugging: IP communication can be used without restrictions in EIP. So newly connected devices can be detected by using the just mentioned SOA-based methods. After configuration, the controller can establish a connection to the new device without interrupting the existing connections.

D. Ethernet Powerlink

Ethernet Powerlink (EPL) can be assigned to the RTE categories B and C. It uses a specialized media access mechanism which can be realized in software allowing the use of standard Ethernet components (category B). The respective hardware variant of EPL corresponds to category C. The latter conforms with real-time class 3.

1) Media access & NRT channel: In EPL the media access is controlled by the Slot Communication Network Management (SCNM): One central Managing Node (MN) sends cyclically *PollRequests* to each of the other devices called Controlled Nodes (CN). The CNs may only send data upon request by the MN. Therefore packet collisions or waiting times in switches cannot occur. Indeed, EPL is the only RTE that uses hubs instead of switches. Since there is no need for collision avoidance and hubs do not analyze the frames, they can perform better performance than switches. Non-EPL devices must not be attached to an EPL network because they would interrupt the communication. However, the NRT channel can be realized by exploiting the fact that all EPL nodes resist in a basic Ethernet mode by default after system start-up which offers IP connectivity. After retrieving the self descriptions, the autoconfiguration service can parametrize the MN and, afterwards, close its connection to the EPL network.

2) Required information: The autoconfiguration service must be aware of the installed devices, their node IDs and the corresponding DDFs (called EDS as in EIP). The installed devices and their IP addresses can again be discovered by using SOA methods. The node ID can be derived from the IP address since each EIP device has an address of the form 192.168.100.xxx where xxx stands for the node ID. As in EIP, the EDS can be identified by Vendor ID, Device Type and Product Code. In EPL, these parameters can be obtained by using the *IdentRequest* command. The similarities of EPL and EIP are due to the fact that both use CANopen as application protocol.

3) Hot plugging: By default, EPL does not support any discovery functions. Also, an IP connection to an unconfigured node is not possible during operation. However, as described in [17], it is possible to detect new a CN when it enters the network with a specific preconfigured "entry-node-ID". This ID is regularly polled by the MN. If a new CN is detected, it gets a new ID by the MN. Afterwards, the CN is reachable by IP traffic which is transferred during the EPL asynchronous phase. At this stage, the new device can be detected by the autoconfiguration service. After configuration, the MN can integrate the new device into the cyclically poll interval without interrupting the isochronous data traffic.

E. Ethercat

Ethercat is a category C RTE and can fulfill the demands of real-time class 3. Although the master can be implemented using standard Ethernet hardware, the slaves require dedicated Ethercat controllers. The basic principle of Ethercat is based on a logical ring topology.

1) Media access & NRT channel: The master sends one Ethernet frame containing the input process data for all devices into the ring. Afterwards, the frame is processed "on the fly" by each slave: it extracts its input data, inserts its output data and forwards the frame to the next device. This process is carried out in hardware which results in low and constant delays. At the end of the ring, the fully processed frame is sent back to the master. On the Ethernet layer all Ethercat devices appear as one single device since they all use the same MAC address. Therefore, an IP connectivity is not given and the autoconfiguration service cannot access the Ethercat devices directly. Instead, it must be connected to a special Ethercat ”switchport”, which tunnels standard Ethernet traffic to Ethercat devices. Therefore, the master allocates virtual MAC addresses to the slaves. However, the Ethercat network must be at least in the state *Pre-Operational*. The autoconfiguration service can configure this state by using default parameters. Afterwards, the Ethercat network is transparent for NRT traffic.

2) Required information: As usual, the autoconfiguration service needs information about the installed devices, the DDFs and the node addresses. SOA-based device discovery can also be used in Ethercat after the *Pre-Operational* state has been established. The node addresses can be allocated based on the node's position in the Ethercat ring. The DDFs are called Ethercat XML Device Description and includes the same parameters as in EPL and EIP since Ethercat also uses CANopen. The devices can be identified by using the device's Slave Information Interface (SII).

3) Hot plugging: New devices connected to the Ethercat network are in the idle state and do not interfere the communication. However, in this state they have no IP connectivity, so they must be discovered by Ethercat-inherent methods. Therefore, the master can recognize new devices by regularly sending broadcast read requests to the SII of the devices. The SII is accessible even if the Ethercat device is in idle state.

VI. Generic RTE autoconfiguration approach

This section will summarize the autoconfiguration procedure based on the findings of section V. To realize the aim of keeping the approach as generic as possible IP-based technologies are intended whenever feasible. In general, the autoconfiguration sequence can be divided into five phases. The respective implementation of each phase is again RTE-dependent:

Phase 1: The first step is the configuration of the NRT channel to ensure the IP connectivity required by phase 2. In all RTEs this requires an address assignment – either by an RTE-specific method (Profinet) or by standard TCP/IP methods (EIP, EPL, Ethercat). Additionally, in Ethercat a specific network state must be prepared.

Phase 2: In the discovery phase the devices attached to the network and their basic characteristics (i.e., Product and Vendor-ID) are detected and afterwards their semantic self descriptions are retrieved. As device discovery and device description are both integral functionalities of SOAs, it seems obvious to exploit this paradigm in the context of RTE autoconfiguration for the implementation of a generic discovery and capability assessment method. Details on the device discovery and description process of both protocols can be found in [11]. It should be noted that only these both SOA aspects are utilized in this work. Other functionalities like service orchestration do not matter here. However the use of SOAs could also be useful when the autoconfiguration approach should be integrated into next generation SOA-based automation systems. Device discovery is not needed in Profinet since the available devices and their basic characteristics are published during the DCP address assignment process.

Phase 3: In this phase RTE-dependent information about the network configuration is collected which is needed in phase 4. The individual steps are described in chapter V for each RTE under the point *required information*.

Phase 4: At this point, all information normally provided by the engineer is available. The autoconfiguration service now has to compute all RTE parameters needed by the RTE controller. At last, the compiled RTE configuration file must be uploaded to the controller. As mentioned in the description of the compilation function block in section IV, the details of this phase are not part of this paper.

Another aspect of section V was the hot-plug capability of the different RTEs. The summarization in table I shows, in which RTE hot-plugging is possible and how new devices can be detected. In general, PN RT, EIP and Ethercat support hot-plugging. However, in PN RT and Ethercat RTE-inherent methods must be used for the detection of new devices. In EPL special precautions at the slave device must be taken in order to enable hot-plugging. In PN IRT no hot-plugging is possible since every network topology change requires a restart of communication.

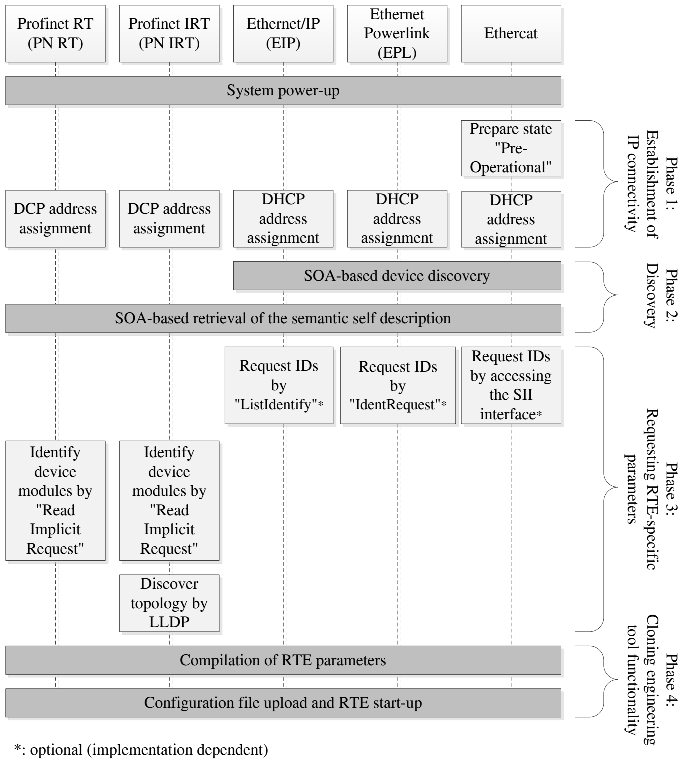
Profinet RT (PN RT)
Profinet IRT (PN IRT)
Ethernet/IP (EIP)
Ethernet Powerlink (EPL)
Ethercat
System power-up
Prepare state "Pre-Operational"
DCP address assignment
DCP address assignment
DHCP address assignment
DHCP address assignment
DHCP address assignment
Phase 1: Establishment of IP connectivity
SOA-based device discovery
SOA-based retrieval of the semantic self description
Phase 2: Discovery
Request IDs by "ListIdentify"*
Request IDs by "IdentRequest"*
Request IDs by accessing the SII interface*
Identify device modules by "Read Implicit Request"
Identify device modules by "Read Implicit Request"
Discover topology by LLDP
Phase 3: Requesting RTE-specific parameters
Compilation of RTE parameters
Configuration file upload and RTE start-up
Phase 4: Cloning engineering tool functionality
*: optional (implementation dependent)

Fig. 5. Autoconfiguration procedure for different RTEs

TABLE I. Hot-plug capability of different Real-Time Ethernets

	Hot-plug capability	Device detection realized by
Profinet RT	+	DCP polling
Profinet IRT	-	-
Ethernet/IP	+	Any IP-based method
Ethernet Powerlink	○	CN with modified ID
Ethercat	+	SII polling

VII. Conclusion and future work

In this paper an analysis of the RTE variants Profinet RT, Profinet IRT, Ethernet/IP, Powerlink and Ethercat with regard to their automatic configuration has been presented. Taking existing approaches into account, general function blocks for RTE autoconfiguration have been derived and it has been checked how their different functionalites could be realized within the mentioned RTEs. In particular, it has been shown how the non-real-time channel of the RTEs can be utilized for device discovery purposes and for the retrieval of semantic self descriptions. The latter are a potential basis for the automatic process data assignment between the process's control application and the field devices.

As part of future work the results of the analysis have to be applied to concrete implementations. Therefore, challenges like the computation of RTE communication parameters as part of the compilation process must be addressed. In the case of Profinet IO there are existing solutions, like in [18] and in [24], which must be merged. Furthermore, the concept for the automatic process data assignment has to be further developed and to be checked for practicability.

Consideration should also be given on the activities of the Time-Sensitive Networking Task Group of the IEEE which aims at the definition of a new industry standard for real-time Ethernet as a successor of Ethernet AVB. There is the chance that the current different RTE variants could be replaced by one common IEEE standard in the future [26]. The impact of time-sensitive networks on the engineering process of industrial automation systems has to be investigated.

Acknowledgement

This work was partly funded by the German Federal Ministry of Education and Research (BMBF) within the project HiFlecs (grant number: 16KIS0266) and the Leading-Edge Technology Cluster "Intelligent Technical Systems OstWestfalenLippe" (it's OWL) (grant number: 02PQ1020).

References

[1] H.-P. Wiendahl, H. A. ElMaraghy, P. Nyhuis, M. F. Zäh, H.-H. Wiendahl, N. Duffie, and M. Brieke, "Changeable Manufacturing - Classification, Design and Operation," *CIRP Annals - Manufacturing Technology*, vol. 56, no. 2, pp. 783–809, 2007.

[2] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren, "Reconfigurable manufacturing systems: Key to future manufacturing," *Journal of Intelligent Manufacturing*, vol. 11, no. 4, pp. 403–419, 2000.

[3] P. Vrba and V. Mařík, "Capabilities of Dynamic Reconfiguration of Multiagent-Based Industrial Control Systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 2, pp. 213–223, 2010.

[4] T. Sauter, "Fieldbus System Fundamentals," in *Industrial Communication Technology Handbook*, R. Zurawski, Ed. CRC Press, 2014.

[5] P. Danielis, J. Skodzik, V. Altmann, E. B. Schweissguth, F. Golatowski, D. Timmermann, and J. Schacht, "Survey on Real-Time Communication Via Ethernet in Industrial Automation Environments," in *19th IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, 2014.

[6] F. Jammes and H. Smit, "Service-Oriented Paradigms in Industrial Automation," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 1, pp. 62–70, 2005.

[7] L. Dürkop, H. Trsek, J. Otto, and J. Jasperneite, "A field level architecture for reconfigurable real-time automation systems," in *10th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2014.

[8] J. M. Mendes, P. Leitão, A. Colombo, and F. Restivo, "Service-Oriented Control Architecture for Reconfigurable Production Systems," in *6th IEEE International Conference on Industrial Informatics (INDIN)*, 2008, pp. 744–749.

[9] M. Loskyll, J. Schlick, S. Hodek, L. Ollinger, T. Gerber, and B. Pîrvu, "Semantic Service Discovery and Orchestration for Manufacturing Processes," in *16th IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, 2011.

[10] G. Cândido, F. Jammes, de Oliveira, Jos Barata, and A. W. Colombo, "SOA at Device level in the Industrial domain: Assessment of OPC UA and DPWS specifications," in *8th IEEE International Conference on Industrial Informatics (INDIN)*, 2010.

[11] L. Dürkop, J. Imtiaz, H. Trsek, and J. Jasperneite, "Service-Oriented Architecture for the Autoconfiguration of Real-Time Ethernet Systems," in *3rd Annual Colloquium Communication in Automation (KommA)*, 2012.

[12] K. Nagorny, R. Harrison, A. W. Colombo, and G. Kreutz, "A formal engineering approach for control and monitoring systems in a service-oriented environment," in *11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013.

[13] A. Girbea, C. Suciu, S. Nechifor, and F. Sisak, "Design and Implementation of a Service-Oriented Architecture for the Optimization of Industrial Applications," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 185–196, 2014.

[14] S. Feldmann, M. Loskyll, S. Rösch, J. Schlick, D. Zühlke, and B. Vogel-Heuser, "Increasing Agility in Engineering and Runtime of Automated Manufacturing Systems," in *IEEE International Conference on Industrial Technology (ICIT)*, 2013.

[15] T. Cucinotta, A. Mancina, G. F. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, and F. Rusinà, "A Real-Time Service-Oriented Architecture for Industrial Automation," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 267–277, 2009.

[16] M. G. Valls, I. R. López, and L. F. Villar, "iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed Real-Time Systems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 228–236, 2013.

[17] G. Reinhart, S. Krug, S. Hüttner, Z. Mari, F. Riedelbauch, and M. Schlögel, "Automatic configuration (Plug & Produce) of Industrial Ethernet networks," in *9th IEEE/IAS International Conference on Industry Application (INDUSCON)*, 2010.

[18] L. Dürkop, H. Trsek, J. Jasperneite, and L. Wisniewski, "Towards autoconfiguration of industrial automation systems: A case study using Profinet IO," in *17th IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, 2012.

[19] L. Dürkop, J. Imtiaz, H. Trsek, L. Wisniewski, and J. Jasperneite, "Using OPC-UA for the Autoconfiguration of Real-time Ethernet Systems," in *11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 248–253.

[20] K. Krüning and U. Epple, "Plug-and-produce von Feldbuskomponenten: Allgemeines Framework dargestellt am Beispiel Profinet IO," *atp edition*, vol. 55, no. 11, pp. 50–56, 2013.

[21] T. Skeie, S. Johannessen, and Ø. Holmeide, "Timeliness of Real-Time IP Communication in Switched Industrial Ethernet Networks," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 1, pp. 25–39, 2006.

[22] J. Jasperneite, "Echtzeit-Ethernet im Überblick," *Automatisierungstechnische Praxis (atp)*, no. 3, pp. 29–34, 2005.

[23] EtherCAT Technology Group. (2014) Industrial Ethernet Technologies: Overview. [Online]. Available: http://www.ethercat.org/download/documents/Industrial_Ethernet_Technologies.pdf

[24] L. Wisniewski, J. Jasperneite, and C. Diedrich, "Effective and fast approach to schedule communication in PROFINET IRT networks," in *IEEE International Symposium on Industrial Electronics (ISIE)*, 2013.

[25] J. Morse. (2012) The world market for Industrial Ethernet Components. [Online]. Available: http://www.iebmedia.com/?id=8595&parentid=74&themeid=255&showdetail=true

[26] S. Kehrer, O. Kleineberg, and D. Heffernan, "A comparison of fault-tolerance concepts for IEEE 802.1 Time Sensitive Networks (TSN)," in *19th IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, 2014.