Mechatronics 34 (2016) 72–77

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Self-reconfiguration of real-time communication in cyber-physical systems

Jan Jatzkowski *, Bernd Kleinjohann

Cooperative Computing & Communication Laboratory (C-LAB), University of Paderborn, Fuerstenallee 11, 33102 Paderborn, Germany

ARTICLE INFO

Article history:
Received 16 October 2014
Revised 10 February 2015
Accepted 19 April 2015
Available online 6 May 2015

Keywords:
Cyber physical systems
Distributed embedded systems
Dynamic reconfiguration
Plug-and-Produce
Real-time communication
Real-time systems

ABSTRACT

Today, in domains like automation and robotics systems consist of various sensors and computation nodes. Due to the temporal dependency in quality of measured data, such Cyber-Physical Systems (CPS) commonly have real-time requirements on communication. In addition, these systems shall become more flexible and scalable, e.g., by adding new components to the CPS. This would be most suitable if a CPS could react to the presence of a new component and reconfigure itself to run afterward with the new component integrated to the CPS. This capability is covered by the term Plug-and-Produce. In this paper, we propose a concept to enable Plug-and-Produce within a CPS whose network uses different communication media, e.g., Ethernet and CAN. To enable real-time communication provided by different communication protocols, their different synchronization mechanisms have to be combined to get a common time base within the entire system. For this purpose, we consider Ethernet as well as CAN-based real-time communication protocols and their synchronization mechanisms. The proposed concept for self-reconfiguration aims to be integrated into our three layered software architecture that is presented as well.

1. Introduction

Nowadays, interaction between a variety of system components controlling physical entities becomes more and more important. In [1], Lee and Seshia define that Cyber-Physical Systems (CPS) are about the intersection of these physical and cyber entities. Depending on the application domain, these systems require at least partial real-time communication and in case of dynamically adaptable systems, this even implies reliable reconfiguration of the communication, especially real-time communication. Although such reconfiguration is already manually practicable, this process is error-prone and strongly time consuming due to user interaction. To overcome these issues, self-reconfiguration of such systems is required. In automation domain, e.g., self-reconfiguration after modification of a production line promises not only to be less error-prone and time-consuming, but also enables a higher utilization of available system components and cost reduction. To enable self-reconfiguration of real-time communication within CPS, we present a three layered software architecture based on the ISO/OSI reference model: Each node of the networked system has an application, middleware, and connectivity layer. While elements of the application layer represent particular functionality of a system component's application domain, the connectivity layer provides the interfaces to the communication channels provided by the hardware of a system component. The middleware layer connects the application and connectivity layer and is responsible for managing inter-node communication, i.e. communication between applications mapped on different nodes of a networked system. A first description of the corresponding concepts is given in [2]. To enable real-time capabilities within a system utilizing different communication protocols without violating these protocols' specifications, we have to combine their communication mechanisms to establish a common time base. Therefore, the presented middleware does also cover time synchronization in such a Heterogeneous system. Due to this, we further provide an overview of particular synchronization mechanisms used by considered communication protocols.

In this paper, we focus on the middleware layer of our architecture, especially on real-time communication and its automatic reconfiguration. Real-time communication is usually realized by a time-triggered approach, i.e. time slots are assigned to nodes that are only allowed to send information within these assigned slots. The proposed self-reconfiguration concept considers modifications like integration of new components to the overall system,

* Corresponding author.
E-mail addresses: jan.jatzkowski@c-lab.de (J. Jatzkowski), bernd.kleinjohann@c-lab.de (B. Kleinjohann).

http://dx.doi.org/10.1016/j.mechatronics.2015.04.014
0957-4158/

i.e. Plug-and-Produce capabilities. Safe modifications are realized by means of pre-defined time slots that are reserved, e.g., for introduction of new system components into the CPS. When such a modification is registered by the CPS, a self-reconfiguration process is started within the middleware layer. During this reconfiguration, our approach considers the required communication from application layer and creates a new assignment of time slots to system components guaranteeing required real-time capabilities. While the self-reconfiguration process itself does not need to provide real-time capabilities, its result has to guarantee the real-time requirements of the new system configuration.

Thus, the main contribution of this paper is the description of our concept for reconfiguration of real-time communication within a CPS by utilizing a common time-triggered communication mechanism that abstracts from real-time protocol specific characteristics. This way, our middleware will enable handling real-time communication requests from application software resided in application layer without knowledge of the real-time communication protocols provided by the connectivity layer that manages hardware interfaces. In addition, to enable Plug-and-Produce capabilities our concept is designed to have particular slots reserved for registering new components added to a CPS.

The remainder of this paper is structured as follows. In Section 2, fundamentals about Plug-and-Produce as well as real-time communication are presented that are used to compare the presented concept with related work. Then, related work is described in Section 3. Our concept for self-reconfiguration is presented in Section 4. Here, we will discuss time synchronization of particular communication protocols as well. This way, we can identify capabilities to synchronize different protocols and establish a common time base. Finally, the paper is closed by a conclusion and outlook in Section 5.

2. Fundamentals

This section provides fundamentals about concepts of Plug-and-Produce and real-time communication. This will support the analysis of related work in Section 3 as well as classification of the proposed concept for self-reconfiguration presented in Section 4.

2.1. Plug-and-Produce

Plug-and-Produce is known from domains like automation and robotics. It is based on the Plug-and-Play technology that originally was developed for general purpose computers as used in office applications and is known, e.g., from the commonly used Universal Serial Bus (USB). Due to the domain-specific requirements of automation and robotics, the term Plug-and-Produce was introduced by the EU funded project SMErobot [3]. In [4], Naumann et al. focus on robot cells at shop floors and define Plug-and-Produce as the ability to add devices to a robot cell and to use the functionality of these devices without the need of configuration. Based on this definition, they define three Plug-and-Produce layers:

Application Offers automatically services to the user depending on the available functionality.
Configuration Configures default values, bandwidth requirements, etc.
Communication Deals with communication protocols and provides, e.g., discovery and addressing of devices.

These layers are hierarchically ordered: Plug-and-Produce on Application layer requires Plug-and-Produce on Configuration layer; Plug-and-Produce on Configuration layer requires Plug-and-Produce on Communication layer.

To compare our concept with state-of-the-art approaches and highlight differences more precisely, we use an additional way to classify Plug-and-Produce respectively Plug-and-Play implementations. This classification depends on the system performance and is given by Zimmermann et al. [5]:

Cold The entire system is shot down, new components are connected and finally, the system is switched on again. Reconfigurations needed due to the modifications of the system are processed during start-up phase.
Hot Components are added to or removed from the system during runtime. However, running applications must not be disturbed by this process.
Coordinated Adding and removing of components is user or program controlled. This way, modification of a system is no longer enabled at an arbitrary point in time, but rather announced to the system. Thus, running applications cannot be disturbed by a randomly occurring reconfiguration process.

These two complementary ways to classify Plug-and-Produce approaches will be considered for related work in Section 3.

2.2. Real-time communication

Due to the interaction between physical and computational entities within a CPS, timing behavior of communication is important for correctness of results produced by a CPS. Not only the measurements of sensors and processing of theses measured data but also the transmissions of these data have to be finished before a pre-defined deadline to achieve correct behavior of a CPS. Therefore, Kopetz states that real time is an integrated part of the real world that cannot be abstracted away [6]. In [7], Buttazzo distinguishes three levels of real-time depending on the consequences of missing a deadline: hard, firm, and soft real-time. In soft real-time systems, missing a deadline causes performance degradation, but results can still be useful for the system. In contrast, results of a firm real-time system are useless when missing a deadline and in hard real-time systems, missing a deadline may even cause catastrophic consequences on the controlled system. Consequently, different real-time systems can differ with respect to the consequences of missing a deadline although all of them are labeled real-time.

With respect to real-time communication, Kopetz defines a set of requirements including timeliness and flexibility [8]. Timeliness covers among others clock synchronization, i.e. a global time base is required. This way, observations like sensor measurements are guaranteed to be temporally relevant for the controlled system when processed by a computation node. Flexibility refers to the adaptation capabilities of real-time communication in case of different supported system configurations that can change over time. It should be possible to add new sensors and/or computation nodes to a CPS without violating temporal guarantees of the original CPS. However, flexibility is limited by the bandwidth available by the given communication channels.

In the automation and robotic domain, communication is often based on Ethernet as well as Controller Area Network (CAN) as protocols to realize physical and data link layer of the standardized OSI 7-layer reference model. Ethernet and CAN both are capable of event-triggered communication. Since events can occur at an arbitrary point in time, i.e. randomly, no temporal guarantees can be provided in general. For instance, two sensors want to transmit their data at the same time via the same communication channel to the same computation node. In case of Ethernet, this would result in a collision that has to be handled by the protocol. In case