SURVEY ON SELF-RECONFIGURATION IN CYBER-PHYSICAL SYSTEMS

Tarannum Kumasgi[1], Prof. Kiran Patil[2]

[1]Student, [2]Assistant Professor, Dept. of Computer Science Engineering,
BLDEA's College of Engineering and Technology, Bijapur, Karnataka, (India)

ABSTRACT

This paper contains a survey on real-time communication in Cyber-Physical Systems (CPS).Due to the requirement of sensors and computation nodes in Robotics and Automation domains; we need a self-reconfigured nature of CPS. Therefore, a self-adaptive system may reduce deployment and maintenance cost by adjusting its own configuration based on the required QoS. This would be most suitable if a CPS could react to the presence of a new component and reconfigure itself to run afterwards with the new component integrated to the CPS. To achieve this capability called Plug-and-Produce is enabled in CPS to support reconfiguration, which makes use of three layered software architecture.

Keywords: *Cyber-Physical Systems, Plug-and-Produce, real-time, self-reconfiguration*

I. INTRODUCTION

Cyber physical system refers to computation (cyber) and physical processes. In CPS, embedded devices/computers and various sensors are interconnected to monitor and control the physical processes[1]. Naturally, the sensing component of CPS is critical to the modeling and management of CPS because it provides real operational data. The goal of sensing is to provide high quality data with good coverage of all components at low cost. In addition, software is embedded in devices, the principle mission of which is not computation alone. CPSs range from relatively small systems, such as aircrafts and automobiles, to large systems such a national power grid.

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[2].

Since 2006, the National Science Foundation (NSF) has awarded large amounts of funds to a research project titled "Science of Integration for CPSs." Many universities and institutes (such as UC Berkeley, Vanderbilt, Memphis, Michigan, Notre Dame, Maryland, and General Motors Research and Development Center) have joined this research project [5][3]. Analogously, in 2004 the European Union (EU) began the Advanced Research & Technology for Embedded Intelligence and Systems (ARTEMIS) project, which aims to tackle the research and structural challenges faced by the European industry by defining and implementing a coherent

research agenda for embedded computing systems [4]. In addition to these efforts, researchers from other countries, such as China and Korea, have begun to be keenly aware of the significance of CPSs research.

II. LITERATURE SURVEY

2.1 General Architecture of Service-Oriented CPS

Unlike conventional embedded systems with concentration on the physical devices, service-oriented CPS is typically designed to co-ordinate the computational and physical parts of a system. Since physical devices typically have limited resources and computation power, it is not always possible to design and execute complex computation and processes. To cope with this challenge, some approaches propose a service-oriented architecture of CPS to handle complex processes using computational and physical resources. The benefits of using service-oriented architecture are twofold:

(i) Provide users a unified access/communication protocol to interact with physical and software elements

(ii) Facilitate to build flexible systems while preserving efficiency and scalability

In general, in the reviewed works, the generic architecture of service-oriented CPS contains three layers: access, service, and application.

• **Access layer:** It facilitates a standard platform for physical devices. First, it provides a networking environment for connecting devices (e.g., RFID, NFC or wireless connection such as IEEE 802.15.4, ZigBee, 6LoWPAN, or Wireless M-Bus). Second, it provides a common representation of physical devices in the system. The data model is used to accommodate various devices and resource data formats. The meta-data relating to physical devices are also included such as spatial attribute, temporal attribute, state representation. Third, it is responsible for device management by detecting and identifying newly connected/disconnected physical devices and their resources.

• **Service layer:** It is a logical abstraction layer on top of the access layer. The main goal of this layer is to cope with the heterogeneity issues of different physical components provided by the access layer and software components. Typically, this layer relies on the Web protocol such as TCP/IP, HTTP. First, service layer provides mechanisms (e.g., adapter, wrapper) allowing to transform physical device to physical service component. It defines the data retrieval for services and device access methods (e.g., push, pull, publish, subscribe). Second, it allows discovering and registering new software components (e.g., computational services). Third, it also provides a common repository of component services to be used in the system. The service repository maintains comprehensive information about published services in terms of functionality, QoS, and context information (e.g., sensor, actuator, temporal, spatial, and device related information). Fourth, it also concerns the QoS issue of the provided services. It ensures the adaptability, composability of services so that the users can compose applications in the application layer.

• **Application layer:** It provides high-level management and interaction with physical and software components. It allows users to create applications with ease as create Web 2.0 mashups (i.e., lightweight, ad-hoc composition). It also offers other features such as searchability, sharing, composition, reuse of created application. Typically, the application layer contains a development environment and a runtime environment for creating and executing applications, respectively.

Different approaches for (re-)configuration of networks exist in literature, each focusing on different aspects of the problem.

Lim and Iqbal [6] proposed a middleware containing entities called agents and actors. These entities move across the network of sensor nodes facilitating the adaptive load balancing, monitoring, and activating resources. Lin et al. views and models CPS as a multi-agent system, where each local agent communicates with other agents[7].

Kim et al. [8] includes a network of cyber-nodes that provide computing resources. Each cyber-node has its own knowledge base and database to be shared among the network. This topology encourages scalability of the system as the computing elements and physical devices can be incrementally added without little interference with other elements. Also, distributed topology has no bottle neck point in the architecture so that it can minimize the network congestion. However, since these physical devices have limited resources, the limitation of this topology is that it is not possible to execute complex computation and processes. In addition, the tasks of devices registration, service discovery become harder to manage.

Kyoung and Sang presented a novel information-centric approach for timely, secure real-time data services in CPSs. In this approach, to derive global knowledge of real world phenomena, network-enabled real-time embedded databases communicate with each other, while controlling and communicating with WSNs in a secure, timely manner. Based on the collective information, WSNs are controlled to extract important data directly related to an event of interest. By taking the information-centric approach, nRTEDBs can considerably enhance the efficiency of sensing, while improving timeliness and security.

Kopetz and Bauer et al. developed the time-triggered architecture for real-time communication strongly focusing on fault tolerance, e.g., by means of different kinds of redundancy. Nevertheless, the authors do not address Plug-and-Play functionality, but rather refer to the fact that one of the considered protocols, namely TTP/A, provides such capabilities[2].

Reinhart et al. addressed an automatic reconfiguration of industrial Ethernet networks and presented a five-step model for a coordinated Plug-and-Produce within Ethernet-based networks. But in contrast to real-time communication concept that covers, e.g., CAN in addition to Ethernet communication, they only consider Ethernet as communication medium within a CPS[9].

Marau et al. presented a middleware supporting reconfiguration of real-time networks. Their approach provides hard real-time guarantees and covers hot Plug-and-Play not only by means of adding new components to the system, but also removing of nodes. But in contrast to the real-time communication they focus on Ethernet as communication medium and define their own communication protocol called FTT-SE. Thus, they do not capture standard communication protocols like PROFINET that are currently used in domains like automation and robotics[10].

III. METHODOLOGY

Efforts close to self-adaptive and self-reconfiguration nature particularly address the design and implementation of self-reconfiguration in embedded system software, context-oriented programming, and system-level adaptiveness in CPSs. Works in self-adaptive embedded system software spans phases from requirement engineering to verification. Co design approaches also exist where the hardware/software boundaries blur for greater flexibility.

Unlike work for self-adaptive embedded systems, these solutions focus on a few environmental dimensions, each requiring ad-hoc self-adaptive functionality. In our target applications, complexity arises especially from the combinations that multiple environmental dimensions concurrently generate. Nevertheless, most of these solutions would be hardly applicable in resource-constrained CPSs, due to run-time overhead.

Villegas [11] designed dedicated software support for situation-aware software systems. Villegas approach relies on the end-user as a controller of context management, and these are abstracted in software-based sensor devices. Moreover, whereas self-adaptive approach focus on autonomous systems and can directly deal with physical sensors to acquire context information.

Fleurey et al. [12] present a model-driven approach for creating adaptive firm wares. They model the application as a single state machine and define behavioral variations based on predicates defined over the application state. When such predicates are found true, the system accordingly adapts the state machine transitions.

3.1. Plug-and-Produce

Plug-and-Produce method 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™ [13].

In paper[14], 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:

a. Application: Offers automatically services to the user depending on the available functionality.

b. Configuration: Configures default values, bandwidth requirements, etc.

c. Communication: Deals with communication protocols and provides, e.g., discovery and addressing of devices.

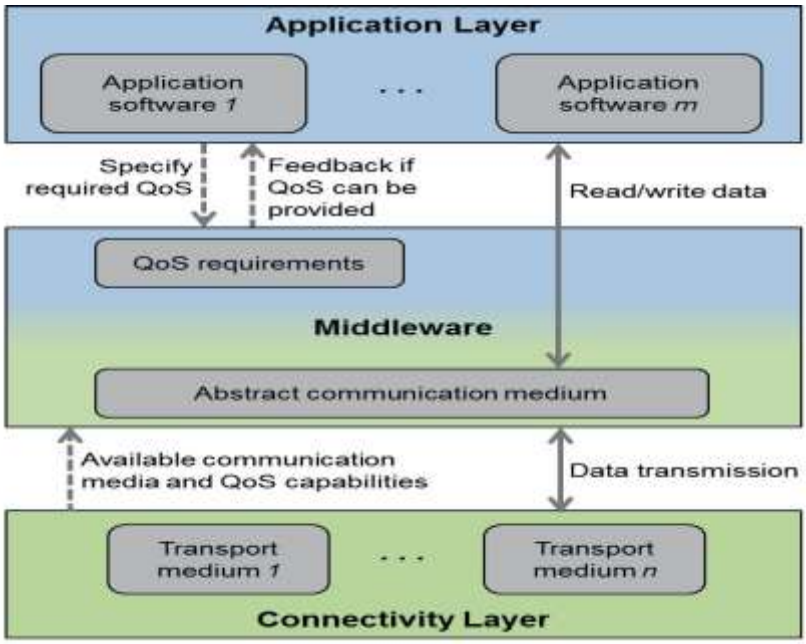

Figure 1. Plug-and-Produce Architecture

3.2. Self-Adaptation and Reconfiguration

The cyber-physical system should be able to modify its behavior according to changes in its environment, errors happening at runtime, and changes to its requirement, i.e., the system should be self-adaptive. A self-adaptive system has the potential to improve its performance or its other QoS parameters, by tailoring the configuration of the system at runtime so as to match the varying requirements of the system to the changing pool of resources that may be available to support those requirements.

Further, a self-adaptive system may reduce deployment and maintenance costs by adjusting its own configuration, based on the required QoS. Thus, a self-adaptive system must continuously monitor changes in its context and react accordingly. Its reaction consists of:

(1) Assessing if an adaptation is required and if so,

(2) Which adaptation is the most suitable answer to a detected change, and finally

(3) Carrying out the adaptation while the system is running.

It is critical that inspite of unexpected changes, the systems are required to operate correctly. Therefore, it is an important task to develop a generic and agnostic approach to express, compare, catalogue and reason about adaptation patterns for coordination in CPSs. An adaptation pattern refers to a reusable abstraction of adaptation strategies, which enables rigorous analysis and formal certification necessary for development of highly trustworthy, self-adaptable cyber-physical systems.

3.3. Abstraction of Communication Media within the Middleware

For real-time capabilities, an abstract cycle-based time-triggered communication medium: Cycles with a fixed duration (cycle length) are specified where each cycle consists of three phases as depicted in Fig 2:

Phase 0 is used for synchronization and thus processes preparations for Phase 1.

Phase 1 is split into slots of predefined equal length to cover real-time communication.

Phase 2 can be used for event-triggered data transmissions, i.e. non-real-time communication.

Most real-time capable communication protocols are based on time-triggered approaches, i.e. these protocols have a similar structure to the proposed abstract communication medium shown in Fig 2. Consequently, on the one hand, parameters of the common abstract communication medium – e.g. cycle length, length of the different phases, and slot length within Phase 1 – have to depend on the real communication media within a CPS. On the other hand, the global time base that is required for real-time communication within a CPS can be established based on the synchronization mechanisms already implemented by the underlying communication protocols. This supports establishing a global time base, but it will also cause additional jitter due to the need of synchronizing time bases of different protocols that already include jitter caused by the protocol specific synchronization.

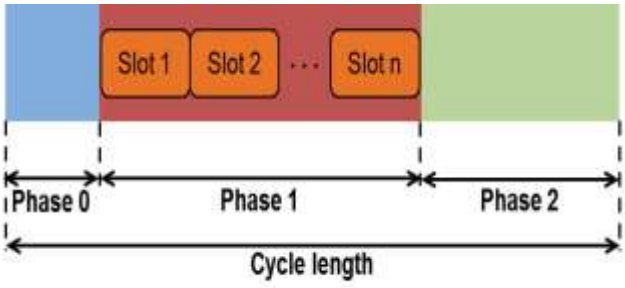

Fig 2. Cycle of the Abstract Communication Medium

3.4. Self-Reconfiguration to Enable Plug-and-Produce

A coordinated Plug-and-Produce [12] is an integration of a new component to the running CPS is processed at a point in time that is controlled by the running CPS itself. This way, we can avoid to disturb the running CPS during its reconfiguration process. The presence of a new component within a running CPS is noticed as follows: One slot within Phase 1 of the abstract communication medium is reserved for registration of a new component during runtime of the CPS. This way, we can guarantee that adding a new component can be noticed by the CPS within each cycle of the abstract communication medium. A newly added component has to signal its presence depending on the transmission paradigm of the underlying communication medium it has been added to.

Consider the case that a new computation entity for processing sensor data needs to be integrated into the CAN network of the first robot cell. Due to the publish-subscribe communication model used by CAN, a component newly connected to the CAN bus is enabled to read all sent messages. Consequently, the middleware of the added component is able to access synchronization messages transmitted during Phase 0 and synchronize itself with the CAN-based sub-system. Based on the established global time, the new component can send a self-description within the slot of Phase 1 that is reserved for registration of new components. The self-description of a component that is send at first contains information about the provided functionality as well as required and provided data. Since data size of the self-description depends on provided functionality and data dependencies of a component, transmission of these data can require more than one slot. In such a case, transmission of self-description data is continued at the next cycle using the reserved slot for recognition of new components.

IV. TECHNICAL CHALLENGES

Since many critical challenges need to be addressed in order to accomplish requirements. There are two types of approaches have motivated considerable research to address issues in communications, information acquirement and dissemination, knowledge discovery, resource allocation and management, heterogeneous system integration and asynchronous control, some of the challenges are given below.

4.1 Communication Issues

Many-to-many information flow and opportunistic connection are inevitable in emergencies. Considering a fire emergency, to find safe paths, sensing information may need to be conveyed from many sensors to many mobile evacuees. This will obviously be more difficult, since communications may break down and evacuees will move to escape.

4.2 Information Acquirement and Dissemination

Cross-domain sensing and heterogeneous information flow is inherent features in an emergency response system. To guarantee the safety of people, information in different domains must be acquired (e.g., ultrasonic sensors for localizing people, temperature and gas sensors for identifying hazards, camera sensors for counting civilians and life detectors for searching civilians). These features will raise a challenge to acquire and disseminate information in an efficient way.

4.3 Knowledge Discovery

Partial information and dynamic changes are inherent in an emergency. In such a rough environment, feasible and quick response must rely on data analysis technologies to extract knowledge from sensing data (e.g., counting, discovery, localization and tracking of civilians). Moreover, dynamic prediction and forecast of environmental changes should be conducted to avoid unnecessary casualties.

4.4 Resource Allocation and Management

Limited resources make timely response more difficult. Unlike other sensor-aided applications, the needs of intelligent actuation, scheduling and efficient resource allocation will increase in emergency response systems. Intelligent scheduling is needed to select the best action, while scarce resources must be allocated efficiently to perform actions.

V. CONCLUSION

A solution to provide design-time and programming support for self-adaptive software in resource-constrained Component-based CPS software has been presented by authors. But in this domain, there was a lack of principled design approach and the rudimentary programming environments result in entangled implementations. Thus, a concept for self-reconfiguration of real-time communication within a CPS that can be composed of sub-systems using different communication media, e.g., Ethernet and CAN has been proposed. This proposed software architecture supports coordinated Plug-and-Produce functionality based on a common abstract cycle-based time-triggered communication medium which ensures that adding a new component can be registered by the CPS within one cycle. Authors have presented how the registration of a new component can be processed during runtime when added to an Ethernet sub-system as well as a CAN sub-system.

REFERENCES

[1] Dat Dac Hoang, Hye-Young Paik Chae-Kyu Kim. Service-Oriented Middleware Architectures for Cyber-Physical Systems.IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.1, January 2012.

[2] Kopetz H, Bauer G.The Time-Triggered Architecture.in Proceedings of the IEEE;2003.

[3] J. Sprinkle, U. Arizona, and S. S. Sastry, “CHESS: Building a Cyber-Physical Agenda on Solid Foundations,” Presentation Report, Apr. 2008.

[4] Available at: http://www.artemis.eu/.

[5] Available at: http://newsinfo.nd.edu/news/17248-nsf-funds-cyber-physical-systems-project/.

[6] Iqbal M. and H.B.Lim. A Cyber-Physical Middleware Framework for Continuous Monitoring of Water Distribution Systems. in Proceedings of SenSys '09. 2009.

[7] Lin, J., S. Sedigh, and A.R. Hurson. An Agent-Based Approach to Reconciling Data Heterogeneity in Cyber-Physical Systems. in Proceedings of IPDPSW '11. 2011.

[8] Kim, M., et al. An Application Framework for Loosely Coupled Networked Cyber-Physical Systems. in Proceedings of EUC '10. 2010.

[9] Reinhart G, Krug S, Hüttner S., Mari Z, Riedelbauch F, Schlögel M. Automatic Configuration (Plug & Produce) of Industrial Ethernet Networks. in 9th IEEE/IAS Conference on Industry Applications; 2010.

[10] Marau R, Almeida L, Sousa M, Pedreiras P. A Middleware to Support Dynamic Reconfiguration of Real-Time Networks. in IEEE Conference on Emerging Technologies and Factory Automation (ETFA); 2010.

[11] N. Villegas. Context Management and Self-Adaptivity for Situation-Aware Smart Software Systems.PhD thesis, University of Victoria,Canada, 2013.

[12] F. Fleurey et al. A model-driven approach to develop adaptive firmwares. In Proc. of the 6th SEAMS, 2011. [13] Jan Jatzkowski and Bernd Kleinjohann.Towards Self-reconfiguration of Real-time Communication within Cyber-physical Systems. Procedia Technology, Volume 15, 2014, Pages 54-61.

[14] Naumann M, Wegener K, Schraft RD. Control Architecture for Robot Cells to Enable Plug'n'Produce. in IEEE International Conference on Robotics and Automation; 2007.