International Journal of Automation and Computing 8(2), May 2011, 221-235
DOI: 10.1007/s11633-011-0577-1

Self-organization of Reconfigurable Protocol Stack for Networked Control Systems

Chun-Jie Zhou Hui Chen Yuan-Qing Qin Yu-Feng Shi Guang-Can Yu
Key Laboratory of Ministry of Education for Image Processing and Intelligent Control, Department of Control Science and Engineering, Huazhong University Science and Technology, Wuhan 430074, PRC

Abstract: In networked control systems (NCS), the control performance depends on not only the control algorithm but also the communication protocol stack. The performance degradation introduced by the heterogeneous and dynamic communication environment has intensified the need for the reconfigurable protocol stack. In this paper, a novel architecture for the reconfigurable protocol stack is proposed, which is a unified specification of the protocol components and service interfaces supporting both static and dynamic reconfiguration for existing industrial communication standards. Within the architecture, a triple-level self-organization structure is designed to manage the dynamic reconfiguration procedure based on information exchanges inside and outside the protocol stack. Especially, the protocol stack can be self-adaptive to various environment and system requirements through the reconfiguration of working mode, routing and scheduling table. Finally, the study on the protocol of dynamic address management is conducted for the system of controller area network (CAN). The results show the efficiency of our self-organizing architecture for the implementation of a reconfigurable protocol stack.

Keywords: Networked control system (NCS), communication network, self-organization, protocol stack, reconfiguration

1 Introduction

A networked control system (NCS) is a distributed control architecture where sensors, actuators, and controllers are interconnected through a real time network. It is the basis of process control systems and manufacturing systems in the information epoch. The communication protocol stack that directly affects the perceived communication quality of service (QoS) plays a critical role in determining the system performance. However, a unified communication protocol standard is not available at present for the reasons of commercial benefits, history, and multi-application objects[1]. There is a large number of protocol standards for industrial communication at present, and even for the same protocol, differences exist. For example, the control area network (CAN) can be different from the standards of application layers: DeviceNet, CANopen, or user-defined[2]. Based on different standards, industrial automation makers provide various types of devices or systems separately, resulting in problems of distributed infrastructure complexity and communication environment heterogeneity. In addition, the availability of the communication resources may change unexpectedly[3], due to changes in the user demand, or disturbances in the network environments. Consequently, the control algorithm in the system will not render the intended results if certain QoS conditions (e.g., time delay) are not observed. It is inherently difficult to guarantee punctuality and predictable QoS because the processing overhead of protocol stack is changeable with different hardware or software implementations, and different QoS control objects of heterogeneous protocols are hard to be coordinated as a whole when lacking a unified architecture that specifies components and interactions therein.

To cope with this challenge, there has been an increasing emphasis on developing reconfigurable protocol stacks with self-adaptability in such a distributed, heterogeneous, and changeable environment. The protocol stack reconfiguration is to implement a software environment that supports the flexible composition of the protocol components for a predictable QoS level. The reconfiguration behaviors on the corresponding protocol component will perform once environment constraint or system requirement change, such as parameter reassignment, service updating, and functionality replacement. Reconfiguration methods can be classified into two categories: static reconfiguration and dynamic reconfiguration. Although the static manner has been widely applied in some small or soft real time applications with benefits of rapid response and easy implementation, the repeated reconfiguration interruption during program running would cause a serious degradation on system performance if the environment changes frequently. Dynamic reconfiguration focuses on performing a self-organized process that makes the system converge toward desired beneficial structures or functions while avoiding interrupts during the program execution[4]. In practice, the general way of realizing deadlock-free dynamic reconfiguration is to decouple the interdependency between system resources, implement reconfiguration for each functional segment and develop self-organized methods for interaction and coordination in the distributed structure[5]. In this paper, a novel architecture is designed to implement a hybrid static and dynamic reconfiguration scheme with the emphasis of

Manuscript received May 3, 2010; revised July 18, 2010
This work was supported by National Natural Science Foundation of China (No. 60674081, No. 60834002, No. 61074145).

the self-reconfiguration implementation by managing the information exchanges inside and outside the protocol stack.

Traditional monolithic protocol stacks are static in nature[6]. It potentially hinders the networking of products from different manufactures or standards, let alone the environment adaptability. Because of this problem, researchers in the networking community have begun to focus more efforts on the design of a reconfigurable protocol stacks since the 1990s. The concept of dynamic configuration of the protocol stack was introduced by Muhugusa et al.[7], which is an environment that lets applications dynamically mix and match protocol functionalities according to their requirements and network availability. The work of Muhugusa et al.[7] presented a hierarchical framework for constructing adaptive protocol stacks by replacing the entire protocol stack with a new one. The architecture implemented as a separate micro-protocol module was given in [8, 9], which supports the framework for constructing configurable protocol and services. Seng et al.[10] discussed issues of generating the reconfigurable protocol stack. Through its new design model, a stack could dynamically establish itself with any updated specification simply by reading in the appropriate information. However, these researches focused on the mechanisms of enacting a reconfiguration process, mainly implemented in a pre-planned manner, and many functions are still centrally organized and required significant manual configuration for the deployment and operation.

In order to improve the system self-adaptability, self-organization theory and methods were developed to reduce the administration effort of users or network designers and enact the reconfiguration manner. A self-organizing design can be characterized with behaviors of local interaction and implicit coordination[11]. It means the coordination information among components is not communicated explicitly in a request-response manner, but is inferred form its local information. Based on it, the manager is not searching for entire system space information to obtain the best solution, but is instead building on partially successful solutions to achieve fast convergence to stable state. The self-organized reconfiguration design has become a trend with the emergence of IP-based networks, for example, decentralized congestion control and address auto-configuration. This trend is further accelerated by the advent of ubiquitous computing, where wireless technologies interconnect an increasing number and diversity of devices. The protocol wireless local area network (LAN) and IEEE 802.11 allow devices to spontaneously form cells even when there is no infrastructure access point to provide central coordination of communications. Similarly, the private area protocols Bluetooth and Zigbee support spontaneous network formation to a certain extent. Protocols considered in the Internet engineering task force, mobile ad hoc networks (IETF MANET) Working Group support the spontaneous formation of routing in the absence of centralized routing control[12−15]. To our regret, these practices mentioned above pay little emphasis on the critical real time demand of industrial communication. Yang et al.[16] applied adaptive coded modulation schemes to a wireless networked control system and gave the method of analyzing delay bounds for system stabilization. However, in engineering the communication code, there is still a lack of discussions of the implementation of a reconfigurable protocol stack for NCS at the level of software architecture.

The main contribution of this work is to propose a new layered architecture for the protocol stack that accommodates various communication purposes in NCS with dynamic reconfiguration capability, therein, a self-organization structure is employed as the foundation to manage interaction and coordination behaviors inside and outside the protocol stack. Users can obtain instantiations of the reconfigurable protocol stack for different real time application requirements from the proposed architectural design. Specifically, our architecture is originally derived from the reduced open system interconnect (OSI) referenced model, which has been the common practice for protocol standards in the factory network community[17], e.g., standards of Fieldbus or Industrial Ethernet defined by the international electrotechnical commission (IEC) community. Compared with the existing layered stack models, the routing and scheduling scheme related to the QoS control are highlighted as the core of the protocol stack in our architecture. The cooperation of these two schemes can provide a whole resource control at the system level by utilizing the information from the top application layer and the bottom data link and physical layers. In addition to implementation of a layered stack that comprises common functionalities of industrial communication standards, a self-organization structure of three-level feedback adaptation is adopted to characterize the reconfiguration manager against variations: the self-reconfiguration level inside the protocol stack, the parameter-level, and the code-level reconfiguration outside. The self-reconfiguration is to manage the interactions between layers based on the realization of routing and scheduling schemes with self-learning ability. The parameter-level reconfiguration is aimed at the construction of a proper formal model to evaluate and layout the protocol stack. The code-level reconfiguration is to deal with processes of the code propagation in the system and the code swapping in the node. The code-level reconfiguration would not act until a certain event is detected from the protocol stack space, such as changes of application object and system structure. Finally, the implementation self-organizing information exchange for the reconfigurable protocol stack is illustrated by the case of dynamic address management for CAN.

The remainder of this paper is organized as follows. Section 2 analyzes characteristics of communication activities in NCS. Section 3 elaborates the architecture of the reconfigurable protocol stack for NCS, including the communication protocol stack, the reconfiguration manager, and the reconfiguration interface. Based on analyzing the methods of self-organized interaction and coordination, a triple-level feedback framework for reconfiguration mechanisms is illustrated in Section 4. Specifically, the mechanisms of internal and external information exchange are described in detail. Under the proposed design framework, a self-

organizing protocol case in the CAN system is studied in Section 5. Finally, Section 6 gives concluding remarks and future research topics.

2 Communication requirements of NCS

A networked control system is a fully distributed real-time feedback control system, as shown in Fig. 1, where the real time network that connects the devices in the system, namely control network, is usually divided into a number of segments with different application objects. As for the inner segment, Fieldbus protocols (e.g., CAN, Foundation Fieldbus (FF), and Profibus) that determine the type of medium access controller (MAC) are the common practice to deal with the connection of field devices. Additionally, one master node is defined to govern the message scheduling of the inner segment and the dynamic connection to outer segments, while other slave nodes act as functional nodes of the sensor, controller, and actuator which occupy the channel according to the plan of the master node. On the other hand, considering the outer segment, the nodes distributed in different segments communicate with each other through Intranet or Internet, where the TCP/IP protocol suite is widely applied. The Intranet is isolated from the outer world and deployed in a fixed multi-level topology, whereas the topology of the Internet is world wide connected and often deployed randomly. In such a segmented master-slave structure, the data forwarding among segments should be controlled by a routing protocol of QoS guaranteed and the master node is responsible for the reconfiguration of the scheduling table for messages from the outer segment. Thus, all the remote operations on slave nodes are controlled by the master node of each segment.

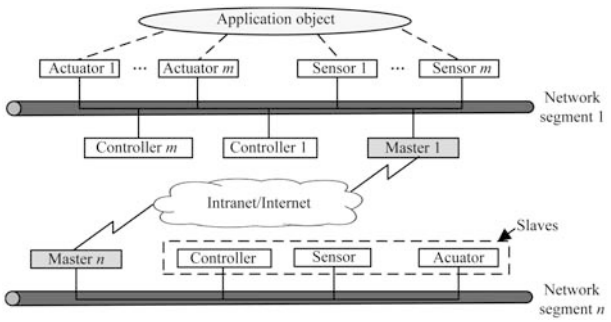

Fig. 1 Typical structure of networked control system

The data transmitted in NCS are classified into three types: periodic data, sudden data, and non-real time data. The periodic and the sudden data are real time data with different time characteristics. The periodic data are generated by sampling and control tasks, which are time critical and their uncertainty would affect the stability of the closed-loop control. The sudden data are commonly issued by the emergency task, which demands the most rapid response time and shortest transmission delay, but the delay uncertainty is not considered.

In such a distributed control system, the protocol interconnection problem under the heterogeneous environment and the real time performance under the changeable communication environment become significant challenges. It calls for the protocol stack of NCS to be capable of reconfiguration ability that supports two types of configuration: the static configuration and the dynamic configuration. On the one hand, in the heterogeneous system environment, various QoS requirements have to be mapped into protocol properties for this selection of protocol components. Hence, for the interconnection problem, the protocol components are composed and downloaded into the node platform in a static manner, which is usually deployed before the system is put into operation. The desired static protocol stack reconfiguration has been achieved by various approaches. For instance, in the X-kernel runtime framework, micro-protocol objects form the functionality of the protocol when assembled according to a graph definition[18]. The Cactus and Appia system extended this concept, proposing a hierarchical composition mechanism for composite protocols based on QoS requirements[19, 20]. On the other hand, in networked systems, the communication channel is affected by lots of uncertainty issues, such as topology change, the traffic load variation and signal interference. These issues represent the offered QoS and express the communication resource availability. The dynamic reconfiguration is to provide a mechanism that lets the protocol stack adapt to the QoS variances through adjusting the parameters, or mixing and matching protocol components at runtime. Although most of the aforementioned frameworks have also declared support for dynamic reconfiguration, they still bear a serious disadvantage: the correctness of composition cannot be completely assured or formally verified. Besides, the aspect of real time reconfiguration is remains unclear on the architectural level. The aim of this paper is the hybrid static-dynamic architecture for the protocol stack reconfiguration of NCS, especially the control structure of the dynamic reconfiguration process. The dynamic process should take the time delay as the primary goal of reconfiguration.

Because of network transmission, the performance of the control system is assumed to be affected by QoS parameters such as delays, jitters, packet losses, and link failures[21]. All the real time data transmissions meet the delay bound, which is the most important performance indicator for NCS. In order to assure good performance, the time delay T_{delay} should be in a certain range, $T_{\text{delay}} \in (T_B, T_C)$. The boundary points ($P_B$ and P_C) are determined by the system stability condition (see Fig. 2)[22]. If T_{delay} is out of this range of sampling period, the system performance cannot be guaranteed. More specifically, the total time-delay is composed of four parts: the sending, waiting, receiving, and transmission delays. The sending and receiving time delays which rely on the software and hardware performance of source and destination nodes are divinable. Comparatively, the waiting and transmission time delays are undeterministic, which depend on the communication activities defined in the protocol stack and available communication resource, e.g., traffic load, bandwidth, transmission paths, etc. Thus,

the waiting and transmission time delays are the key factors to evaluate the performance parameter T_{delay}.

Therefore, the dynamic configuration of the reconfigurable protocol stack in NCS is mainly designed to deal with two issues: the reconfigurable scheduling scheme on various traffic loads inside the segment and the reconfigurable routing scheme on various network resources cross segments. The former, which manages the message sending intervals on the communication channel, has a close relation to the waiting time-delay[23]; the latter, which governs the transmission paths and network topology, determines the transmission time-delay[24]. The scheduling and routing scheme are implemented as a part of the network and transport layer from the viewpoint of the communication protocol stack. In contemporary industrial communication, from Fieldbus to Industrial Ethernet and wireless network, the reduced OSI reference model (omits the presentation and session layer) has been taken as the basic architecture to comprise different protocol entities in a stack (see Fig. 3). The performance routing and scheduling scheme are difficult to predict due to various structures and MAC services provided by different standards. Motivated by the existing architectures, we provide a unified architecture supporting reconfiguration management to accommodate heterogeneous protocol standards and different application objects. The routing and scheduling schemes related to the QoS control are highlighted as the core of the protocol stack in our implementation. It requires cooperation between the routing and scheduling schemes that can provide a whole resource control on a system level by utilizing the information from the top application layer, the bottom data link and physical layers.

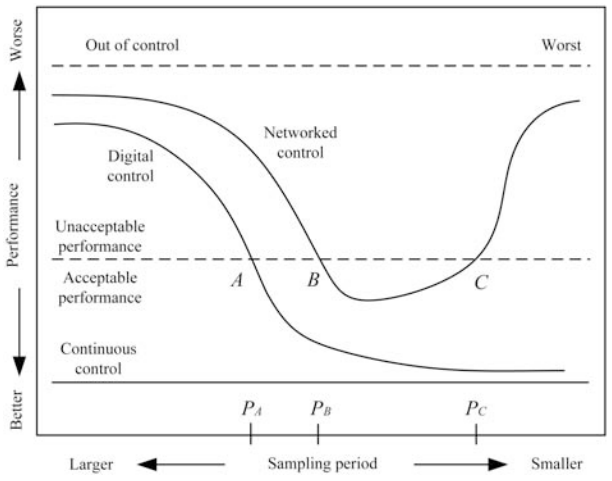

Fig. 2 Performance comparison of continuous control, digital control, and networked control cases

3 Architecture and management of reconfigurable protocol stack

Reconfigurable protocol stack requires software architectures that are flexible and can support tools and algorithms from a variety of sources and domains. The architecture design is the key to implement a reconfigurable protocol stack, which is a specification of protocol component and interface. It lets applications dynamically adjust the parameter configuration of each layer, or mix and match protocol function blocks according to control requirements and network availability. In the master-slave structure, slaves are performed as a reaction to the plan on the master, thus, the protocol stack embedded in slave nodes is usually a reduced version of that in the master node. All the discussions below are concerned with the implementation of a complete protocol stack for the master node in NCS. The architecture and management framework of the reconfigurable protocol stack is illustrated in Fig. 4, which is composed of three parts: the layered communication protocol stack, the formalized reconfiguration manager, and the hybrid static-dynamic reconfiguration interface. It satisfies the following characteristics, and implementation of dynamic reconfiguration is the emphasis of this paper.

1) Formalized reconfiguration engine

It provides mechanisms that keep track of data streams, identify the type of changes, evaluate the reconfiguration result and generate a new stack code accordingly. It is important to determine the performance and efficiency on the proposed reconfiguration schemes, but the field testing is usually quite costly and even not realistic. The formal technology is proved to be effective in validation and evaluation of protocol design by abstracting the protocol behaviors, system requirements and resources availability in a mathematical or graphic manner. It is the key to realize an efficient reconfiguration engine.

2) Hybrid static-dynamic workflow

It provides the flexibility to build the protocol specification statically and adapt the stack configuration dynamically. Such reconfigurations include: i) configuring the properties or composition of protocol components in a static way that the code is directly downloaded into the node with reference to a predefined protocol library; ii) adding, removing or swapping protocol components in a dynamic way that the system operation would not be stopped or interrupted. In general, the workflow of dynamic reconfiguration is more complicated than the static one, considering the utilization of idle bandwidth, the online test of reconfiguration schemes, and so on.

3) Diverse communication channels

They provide communication channels for real time and non-real time applications separately in the protocol stack based on the data transmission characteristics of NCS. The non-real time channel deals with the problem of seamless integration to commercial network to support the monitoring service of control systems, such as FTP or HTTP. Comparatively, real time channel aims at providing a set of real time guaranteed networking and controlling services in coordination with the real time scheduling scheme.

3.1 Communication protocol stack

The layered protocol stack usually consists of five layers:

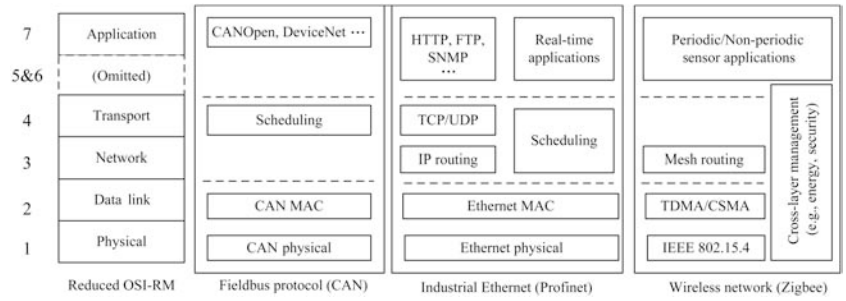

Fig. 3 Typical architectures of the communication protocol stack in NCS (e.g., CAN, Profinet, and Zigbee)

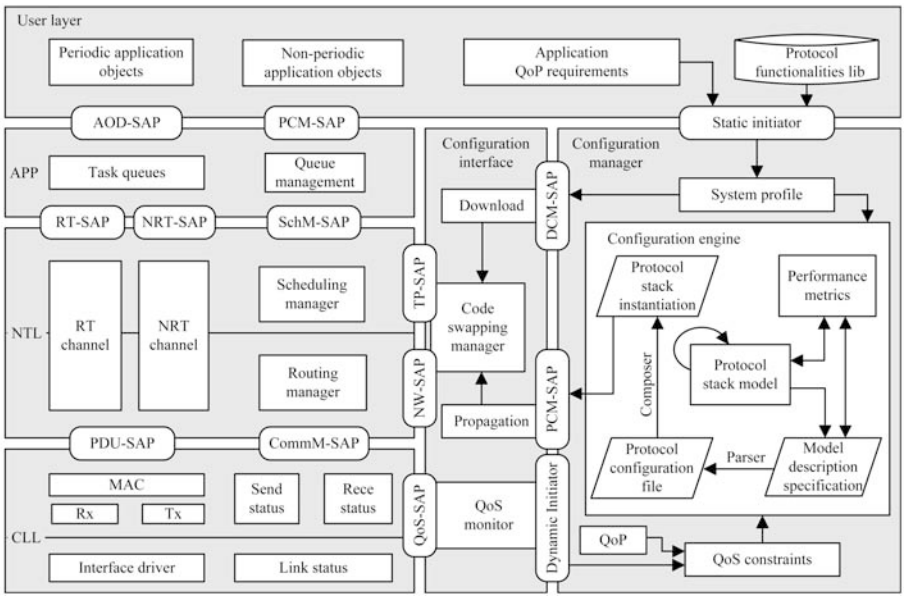

Fig. 4 Architecture and management framework of reconfigurable protocol stack for NCS

the physical, data Link, network, transport, and application layers with reference to the OSI model[17]. Motivated by the existing layering architecture, the function and layer compositions are reconstructed for better severing the reconfiguration viewpoint. It consists of the communication link layer (CLL), the network transmission layer (NTL), and the application layer (APPL). This new architecture is emphasized on utilizing cross layer information to constrain the working of reconfigurable routing and scheduling schemes.

1) Communication link layer

The CLL represents the combination of physical layer (PHY) and the data link layer (DLL). In common, the functionalities of PHY and DLL are integrated in a special-purpose chip. Thus, different from the NTL, the CLL is the emphasis of hardware reconfiguration that is concerned with the appropriate parameter configuration on the hardware chip. Through the parameter configuration, the CLL can provide a reliable point-to-point physical link for the routing service of NTL. The reliable link involves two levels of meanings: the one is to assemble or disassemble the data between the NTL and the physical channel depending on the communication controller; the other is to report the link status and deal with the transmission errors, such as conflict, link loss, and so on.

- Interface driver: The configuration of the initialization information, such as data rate, working mode, frame length, and message validation mode.
- Link status: The indication of the link availability, such as busy and free.
- MAC: The state machine to govern the states of sending, receiving, and resending of nodes, especially in the case of conflict.
- Tx & Rx: They perform as an assistant to the resend function of MAC through the management of queues of sending and receiving data.
- Send & Rece: They match with the fault type when errors are detected during the message checking.
- PDU-SAP: The service access point (SAP) is responsible for reading and writing of protocol data.

- CommM-SAP: The SAP of communication module provides status fetching operations.

2) Network transmission layer

The NTL is mainly composed of three parts: the transport channel, the scheduling scheme, and the routing scheme. They are all carried out with software, i.e., research objects of software reconfiguration. The transport channel is to provide networking and controlling services in coordination with the scheduling scheme. The flexible scheduling scheme is implemented to control the data flow between the APPL and the NTL according to receiving of data transmission request and information of scheduling reconfiguration. Based on the non-periodic transmission declarations or measured QoS values from the network, the scheduling scheme can proactively reconfigure the parameters of scheduling algorithms accordingly, such as bandwidths for the periodic tasks and non-periodic tasks, the polled sequence of slave nodes, or the length of a time slice. On the other hand, after the transmission is triggered, the routing scheme can provide the QoS guaranteed end-to-end data link service for the scheduling scheme. Periodic and non-periodic tasks from APPL can be scheduled with the assumption that transmission delays are all bounded into a certain range once the routing path and the rerouting criteria are determined.

- RT channel: Real time (RT) channel includes the packet assemble/disassemble for period and emergency message tasks. The RT channel forwards the packet from DLL to APPL directly.

- NRT channel: Non-real time (NRT) includes a TCP/IP protocol suite to support non-real time applications, such as the web server and the code propagation. The TCP/IP suite provides a seamless integration to general-purpose commercial protocols, like HTTP, FTP, etc.

- Routing manager: The improvement of the IP protocol to maintain the routing table distributed in nodes with the changes of network topology. It sets the shortest (lowest cost) path for real time messages forwarding between network segments.

- Scheduling manager: It is a flexible scheduling scheme to maintain the scheduling table distributed in each segment with changes of QoS variations and application objects. It governs the division of time slices for both real time and non-real time messages in a single segment.

- RT-SAP: The interfaces for the real time message recognition and the packet assemble/disassemble.

- NRT-SAP: The interfaces for non-real time message recognition and its packet assemble/disassemble interfaces.

- Sch-SAP: The SAP of scheduling module reads the length and type of messages, submitting the scheduling table to configure the tasks and renewing queues of tasks in the APPL.

3) Application layer

The APP layer is the interface between the protocol stack and users to collect the node information and interpret the user task. It provides highly reliable data services for the applications on the user layer. Meanwhile, all the application data calling for the services of NTL are divided into two queues: the periodic and the non-periodic. Once the periodic and non-periodic queues are formed, the data are ready for the network transmission. If nodes are added in or removed from the system during the network transmission, the node management will reassign the node address and update this information for the scheduling scheme of NTL. Besides, when changes in system structure or application object are detected, the system model and the node model will be constructed to re-initialize the configuration of platform resources that are required by the configuration mode of CLL.

- Task queues: It allocates the space for queuing of period, aperiodic real time and aperiodic non-real time message tasks separately.

- Queue manager: It renews and resets the message queues.

- Node manager: It configures the address of slave nodes of the same segment. Commonly, this service is closely related to the implementation of plug & play functionality.

- AOD-SAP: It is the SAP for the exchange of application object data (AOD) between the APPL and the NTL. It submits the message to the APPL, and sends the application data to the NTL according to the scheduling table.

- PCM-SAP: It is the SAP for partial configuration manager, which issues the request to reconfigure the scheduling table to accommodate the changes of QoS level and message queues to a certain extent. The PCM would not change the composition of protocol stack, that is, the relationship of protocol function blocks.

3.2 Reconfiguration manager

The reconfiguration manager is a formalized configuration engine to govern the static and dynamic reconfiguration processes (see Fig. 4). In the reconfiguration manager, the static reconfiguration on the protocol stack can be completed according to the system profile; meanwhile, the configuration information would be reflected in the protocol stack model (PSM) for runtime performance evaluation. On the other side, the dynamic reconfiguration can be completed through the configuration engine, which

mainly consists of PSM, model description specification (MDS), performance metrics (PMs), protocol configuration file (PCF), and protocol stack instantiation (PSI). The PSM and the MDS are both formal descriptions on a specific protocol configuration, while the former focuses on functional verification and the latter is parsed to be a PCF. Before parsing into a configurable file, the performance evaluation should be performed first either on the formal model or specification with PMs. The result of PCF is a platform-independent intermediate representation that describes both the sequence and type of protocol mechanisms. Subsequently, one or more target platform-specific PSI′s may be generated by invoking synthesis tools to "fill-in" and interconnect the necessary platform-dependent mechanisms.

- PSI: The instantiation that is composed of three basic elements — message content, message structure and protocol interactions, according to the three essential factors of protocol — syntax and semantics interaction rules.
- PCF: The guidance file that distinguishes between the instantiations before reconfiguration and after reconfiguration on the elements of message content, message structure, and protocol interaction.
- MDS: It makes the corresponding protocol specification, which refers to the graphic formal model, for code generation and performance evaluation based on the formal language.
- PSM: It builds a graphic formal model to validate protocol properties — liveness, safety, boundedness, and completeness.
- PMs: The QoS level required by application objects, such as time delay, network utilization, and network throughput.
- System profile: The system structure configuration, node role allocation, and control requirement abstraction provide protocol stack code directly for static reconfiguration, as well as the initial states information for dynamic reconfiguration. A system profile may be directly generated by selecting one or more of the persistent configurations or instantiations pre-defined in the protocol functionalities library, when specifying their control requirements via the static initiator interface.
- QoS constraints: The actual and predicted value of QoS, which is obtained from monitoring the changes of network conditions and control requirements.
- Static initiator: It issues the static reconfiguration process during the initiation or running period. Through stopping or interrupting the system, designers or users manually download the new code file which is fetched from the protocol functionalities library into the node.
- Dynamic initiator: It issues the dynamic reconfiguration process. The dynamic reconfiguration is implemented locally without affecting the working of nodes in normal control loops or stopping the system. During the running period, the dynamic initiator would keep track of the actual QoS level from the monitor. When the QoS does not meet the design requirements, a self-organizing code propagation process will start in the network based on results from the reconfiguration engine.
- SCM-SAP: The SAP of static configuration manager for compiling the system profile to be a platform executable file, e.g., HEX file.
- DCM-SAP: The SAP of dynamic configuration manager for assembling the reconfiguration code file in the style of non-real time packet and setting the queues and priorities of reconfiguration task.

3.3 Reconfiguration interface

The reconfiguration interface deals with the interactions between the communication protocol stack and the reconfiguration manager (see Fig. 4). In practice, many more computing resources are required by the reconfiguration manager than the communication protocol stack due to the formal synthesis work (modeling, analyzing, and simulating). Hence, the functionalities of the manager and the protocol stack are distributed on different nodes or platforms.

- Code swapping manager: The management of code space in the protocol stack, especially on sharing, adding, and deleting operations.
- Download: It loads the stack code manfully, orienting to the static reconfiguration.
- Propagation: It assembles the reconfiguration code files in the style of non-real time packet, sets the queues and priorities of reconfiguration task and controls the code propagation in the network, orienting to the dynamic reconfiguration.
- QoS monitor: It keeps track of the node status and communication status on the CPU and network controller separately.
- QoS-SAP: The SAP for collecting the QoS related values runtime.
- NW-SAP: The SAP for locating the position of the network layer in code space and renewing it.
- TP-SAP: The SAP for locating the position of the transport layer in code space and renewing it.

4 Design of self-organization for reconfigurable protocol stack

On the basis of the proposed architecture and management framework of reconfigurable protocol stack for NCS (see Fig. 4), the dynamic reconfiguration ability of the communication protocol stack has been further strengthened. Self-organization is a great concept for building adaptable distributed systems. Until now, many self-organization methods have been developed for communication networks, wireless sensor networks in particular[12, 14, 16]. Nevertheless, self-organization in the design of a reconfigurable protocol stack of networked control systems is still lacking systematic discussion.

In the context of this work, self-organization serves as a general technique to deal with the interaction between individuals and coordination with the environment during the dynamic reconfiguration process. The self-organizing process for the proposed reconfigurable protocol stack is characterized by self-diagnosis on service, self-planning on configuration, self-evaluation on performance and self-stabilization on code. More specifically, in order to realize self-diagnosis, the local behavior rules for each node are designed based on expressing the overall function in terms of a local property. Then, methods of environment observation, conflict detection, and conflict resolution are defined as for the self-planning. Meanwhile, formal description methods are needed to make validation and evaluation on the results of self-planning. Finally, the self-stabilization element controls the convergence of the protocol stack code to an "intended state" regardless of the initial protocol configuration. That means the old protocol code starts to converge to its intended behavior in self-stabilization, after the latest fault or change is detected in self-diagnosis and the protocol stack model is generated in self-evaluation. However, the self-diagnosis, self-planning, self-evaluation, and self-stabilization are usually discussed separately in the aspects of communication activities, code management, and the formal analysis.

Therefore, the integrated control structure is the key to implementing self-organization in the proposed architecture of a reconfigurable protocol stack. It is a dynamic management of the code and formal spaces in coordination with information exchange activities inside and outside the protocol stack.

4.1 Triple-level self-organization structure

A triple-level self-organization structure is developed to control the reconfiguration procedures that react to changes in the network and user environment. The components in the proposed architecture of reconfigurable protocol stack are implemented in nodes, rather than in a high level of system with a global view. As shown in the structure (see Fig. 5), the components of the reconfigurable protocol stack are spatially divided into three groups: the communication protocol stack, the formal space — modeling analysis and the code space — reconfiguration execution. Furthermore, three levels of adaptation are distinguished as follows.

Self-reconfiguration inside (level 1): The self-reconfiguration occurs inside the protocol stack, which refers to the management of inner interactions between layers through the realization of routing and scheduling schemes with self-learning ability. It is the key to determining the self-organization performance, since the design of self-reconfiguration focuses on the structure and behaviors of information exchange inside and outside the protocol stack. The information exchange architectures are detailed in the following sections.

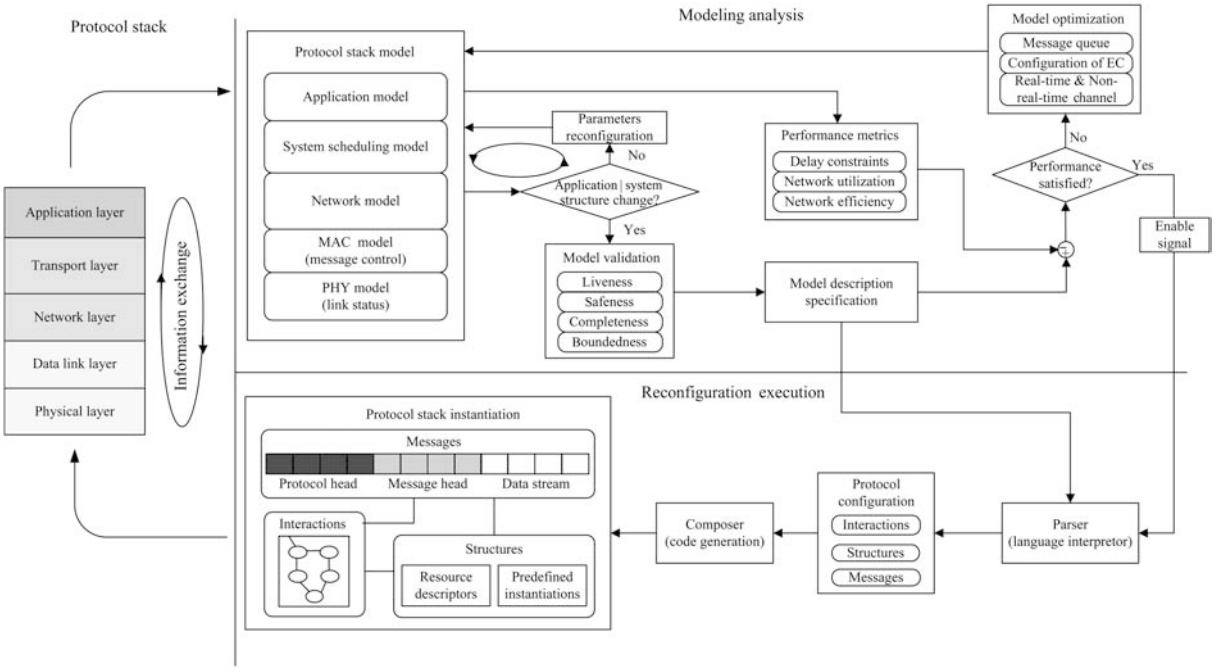

Fig. 5 Triple-level self-organization structure

Parameter configuration outside (level 2): The parameter configuration occurs in the space of modeling analysis, which means to adapt the parameters (e.g., data rate, bandwidth, value of timers, segment size, and queue length) of protocol models for achieving a valid configuration plan for a layer or the whole stack provided that the system structure or application objects are fixed.

Code configuration outside (level 3): The code reconfiguration occurs when changes of system structure and application are detected. It is designed so that it realizes if the changes are so severe that the employed mechanisms of levels 1 and 2 are no longer competitive. It needs the process of model evaluation, code parser, and code composer, through involving both the formal and code spaces. From the composition of protocol stack instantiation, the code reconfiguration is defined as results of the relationship change between protocol components and its impact on message structure and message content. Here, the reconfiguration of message structure means the redefining of an alternative protocol type or network resource. Moreover, the reconfiguration of message content includes the change of message header, which is more complicated than that defined in the parameter configuration.

4.2 External information exchange mechanisms

The protocol stack plays an important role of bridging the node platform and system environment. As shown in Fig. 6, the node platform consists of hardware devices and operating kernel, while the system environment is divided into network environment and user environment.

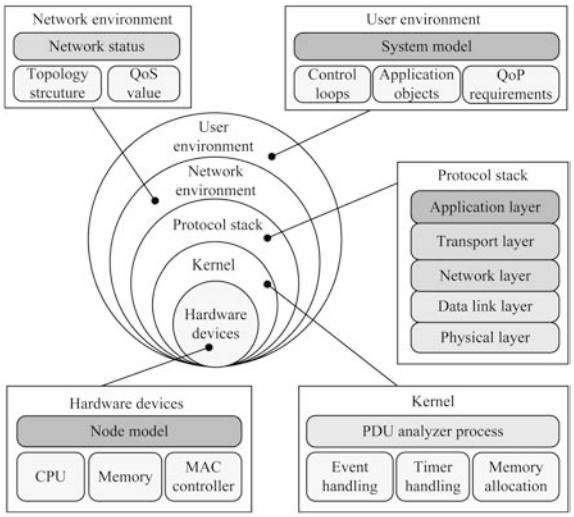

Fig. 6 Architecture of external information exchange

1) User environment

The protocol stack gets the system model from some human-machine interfaces, which comprises the initial configuration of protocol stacks including three basic elements: the information of control loops, the declaration of application objects and the requirement of quality of control performance (QoP). The control loop is to indicate the network scale and message path; the application object contains the information of message length, packet size, and real time type; and the QoP requirement determines the demand on the QoS.

2) Network environment

The network environment is the information of network status that comprises two elements: network topology and QoS value. It is an important outer issue that affects the self-reconfiguration process inside the protocol stack. The network topology is to indicate the change of transmission path and the number of nodes, and based on this variation, the routing scheme will update the routing table of each node through local interactions. The QoS value is to represent the severity of communication interference. It is collected and predicted at runtime as an input to the scheduling scheme, and issues a rescheduling process in the manner of implicit coordination.

3) Kernel

In the node, the protocol stack is only the code segment that deals with the networking functionalities, while the kernel provides services that manage the underlying hardware resources (e.g., CPU, primary and secondary storage, and various I/O peripherals). These kernel services could be used to create a protocol data unit (PDU) analyzer process, which is support for timer handling, event handling, and memory allocation management. Implementing kernel services efficiently is another crucial issue that affects self-reconfiguration performance since all the functionalities of the protocol stack ultimately operate by using these services.

4) Hardware devices

CPU, memory, and MAC are the three main components for the hardware platform, which can be abstracted to be the node model that indicates the service conditions: i) status of timer and interrupt in CPU; ii) usage of static and dynamic space in memory; iii) status of physical link, sending and receiving buffers in the medium access controller.

4.3 Internal information exchange mechanisms

The architecture for internal information exchange inside the protocol stack is described in Fig. 7; especially, the interfaces (e.g., system model, node model, PDU analyzer, and network status) for the external exchange are located in the protocol layers. As is mentioned in the description of triple-level self-organization structure (see Fig. 5), the internal information exchange activities are included in the self-reconfiguration level adaptation. The system model and the node model are represented as the initial conditions for the self-reconfiguration. When the change of system structure or application objects is detected, the system model and the node model would be constructed manually to re-initialize the division of network segments and configuration of platform resources. Here, the internal information exchange activities for self-reconfiguration are characterized by three aspects:

1) Reconfiguration of working mode

Commonly, the operation mode and the fault mode are

the two basic working modes of MAC, except for the configuration mode. The mode switching is controlled by the node self-status and neighbor status. The former is the running condition (e.g., remaining energy and available free space) of the hardware and software itself, and the latter is the sensing result of neighbor nodes (normal or abnormal). Once errors happen during the communication, it will be recorded in the sending or receiving vector of the MAC controller and issue an interrupt signal from the MAC controller to the CPU. After that, the PDU analyzer checks the fault type and starts the interrupt manager to switch the working mode of the MAC controller, from the normal operation to the fault mode. Through the execution of a set of operations, like rollback, step out or reset, the MAC controller will converge to the normal state. The reconfiguration of working mode is to guarantee that the fault node does not interfere the working of others until it becomes normal again.

2) Reconfiguration of routing table

The topology structure in the open network control system is always heterogeneous and changeable. When new devices are added to the system or someone is out of work, the topology of NCS would be changed, then the routing tables distributed in the master nodes are required to be reconfigured to update networking paths for such a new data transmission requirement. Otherwise, the QoS value (e.g., the transmission delay) may increase or decrease unpredictably, and leads to destroy of the system performance. The routing reconfiguration of NCS is a very complicated combinatorial optimization problem, which can be similar to the non-deterministic polynomial complete (NP-complete) problem. Thus, some intelligent algorithms are required to improve the searching ability for the complex solution space. Moreover, the routing reconfiguration of NCS should consider the specific requirements of NCS such as the time constraint. In the previous work[25], an improved mechanism based on genetic algorithm was designed to find the minimum time-delay path for the routing table reconfiguration.

3) Reconfiguration of scheduling table

Roughly speaking, the problem of network scheduling in NCS is to assign a scheduling table (in master nodes) to each transmission entity (non-periodic and periodic tasks in slave nodes) with a network based scheduling algorithm (a set of time and event triggered rules that determine the order in which messages are transmitted). Based on the predicted value of QoS, the scheduling scheme can pro-actively reconfigure the parameter of scheduling algorithms accordingly, such as bandwidths for the periodic tasks and the non-periodic tasks, the polled sequence of slave nodes, and the length of a time slice. Here, considering implicit information from the prediction of QoS, the scheduling algorithms is calculated off-line on and on, but the scheduling table will be updated online only when the predicted and actual QoS are matched. In such a hybrid off-line and on-line operation mode, the cost of reconfiguration of the scheduling table can be controlled at a very low level.

5 Case study

In this section, the dynamic address management for CAN is taken as a case to illustrate the previous self-organization structure that implements the dynamic

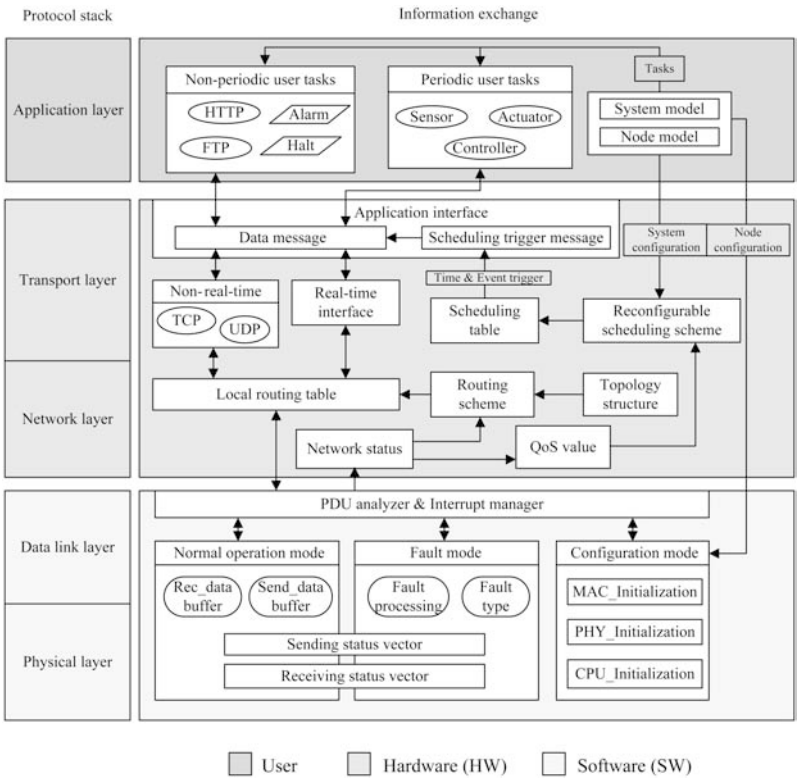

Fig. 7 Architecture of internal information exchange

reconfiguration of the proposed protocol stack architecture. In practice, it is very important, for it is not a good idea to expect workers of industrial fields to do some complicated configuring work, and to achieve the dynamic address management for CANopen, which will make it easier to deploy and maintain the control system. If any slave node fails in the industrial field, it could be replaced by a well-behaved device. After a simple operation in the master node (maybe only by pressing a button), the newly connected node would be configured automatically, without any more work.

The CAN is one of the existing architectures that focus on real time communication under the master-slave structure, and a popular standard that defines physical and data link layers of the protocol and leaves higher layer definitions free to the user. For instance, CANopen is a kind of standard of application layer. In a CANopen-compliant control system, the master node controls all the configuration activities of the slaves through corresponding Node-IDs, including the reconfiguration of the scheduling table and the hardware working mode. The dynamic address management, i.e., automatic Node-ID assignment, is very important to the realization of a self-organization network. In the standard CANopen architecture, the scheduling functionality is defined as a part of the application layer and the network status report can be obtained from the data link layer.

5.1 Protocol implementation

Our proposed architecture of a reconfigurable protocol stack can serve as the foundation of the information exchange for self-organization networking functionalities. It means that the reconfigurable architecture could be various in the implementation of certain protocol components, but should be stable for the interactions among these components. For instance, in this case study, the network transmission layer is added for real time scheduling, and the reconfiguration capability is achieved by a well devised Node-ID management protocol. Moreover, the real time scheduling component is implemented as the core of the protocol stack and reconfigured with an alternative algorithm for the calculation of the scheduling table compared with CANopen at the architecture level.

At present, the Node-ID assigning work can be accomplished by means of layer setting services (LSS) defined in CANopen. However, the standard LSS only provides a mechanism to configure the Node-ID for one slave node, but is not competent for configuring Node-ID for all slave nodes automatically. Here, the proposed self-organizing structure (in the above sections) is to improve the LSS protocol. As a self-organizing process, all the slave nodes can obtain a unique Node-ID through the information exchange between the master and slave nodes. Specifically, as shown in Fig. 8, the Node-ID dynamic management protocol (implemented in the master node) comprises three phases: the first is the preparation phase to count all slave nodes online; the second is the discovery phase to detect the unknown nodes; and the third is the binding phase to configure the Node-ID.

The assigned node-IDs are equal with the slot numbers which reflect logic positions of slave nodes in the control system, so a node-ID also reflects the logic address of a physical slave node in fact. Details of the dynamic address management protocol are shown as follows (see Fig. 9).

First, in the preparation phase, the master detects each slave node with node guarding service. The detection is achieved by the master broadcasting a node guarding frame to the slaves. Then, the switch mode global frame is used to switch all LSS slaves into operational mode, and the switch mode selective frame is used as a response to the request frames to put the selected slave nodes into configuration mode. Next, the switch identification state selective frame is sent to slave nodes, of which the ones already registered in the master node are transferred to be identified.

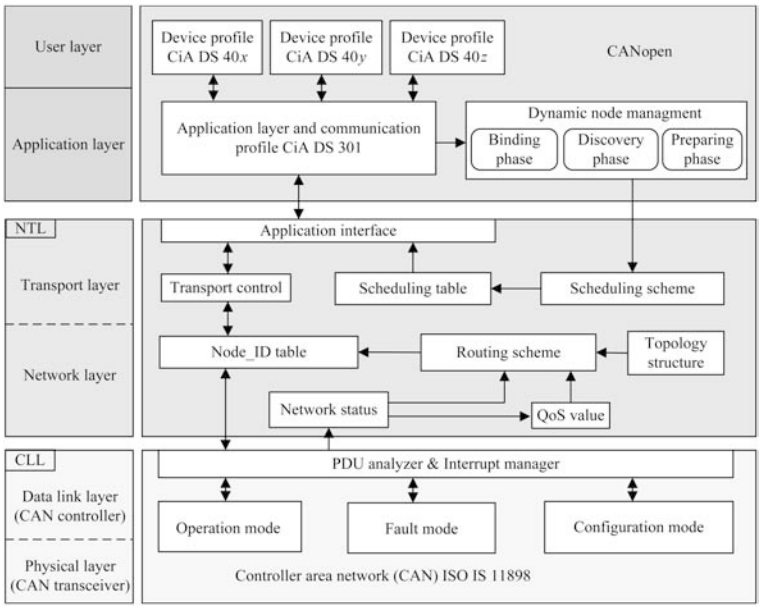

Fig. 8 Reconfigurable protocol stack with dynamic address management

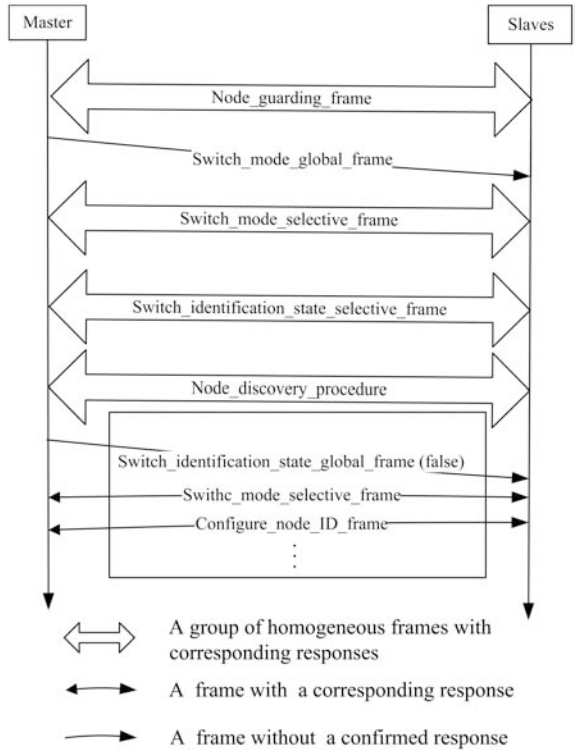

Fig. 9 Time sequences in exchanges between the master and the slave nodes

Second, in the discovery phase, the node discovery procedure is defined to comprise a sequence of identification phases. It is an improvement of the work discussed in [26], which can be used to detect all slave nodes and adopts standard LSS address instead of the 6 bytes self-defined address. Specifically, each of these phases contains a request initiated by the master, followed by different numbers of responses from the slaves. The master node repeats inquiries to the attached devices with more and more precise criteria, until a single node responds and, thus, it is identified. Such procedures will be repeated until all nodes become known in the network. The discovery is based on four new kinds of unconfirmed services: identify unknown remote slaves (IURS), identify unknown slave (IUS), directly identify remote slave (DIRS) and directly identify slave (DIS). These four services form the node identification procedure. As shown in Fig. 10, sometimes, it is necessary for the master to be able to set the state of nodes back to "unknown" or "identified" directly (for example, to force a new detection of devices).

Third, in the binding phase, the node-ID assignment relies on a particular sorting algorithm on LSS addresses. After note-IDs have been pre-assigned to each newly discovered node in the address table of the master node, the LSS configure node-ID service is used to configure all the recent discovered nodes indeed. The switch mode global service and switch mode selective service are used alternatively to ensure only one slave node to be in configuration mode at a time, so as to conform to the constraints of the LSS configure node-ID service.

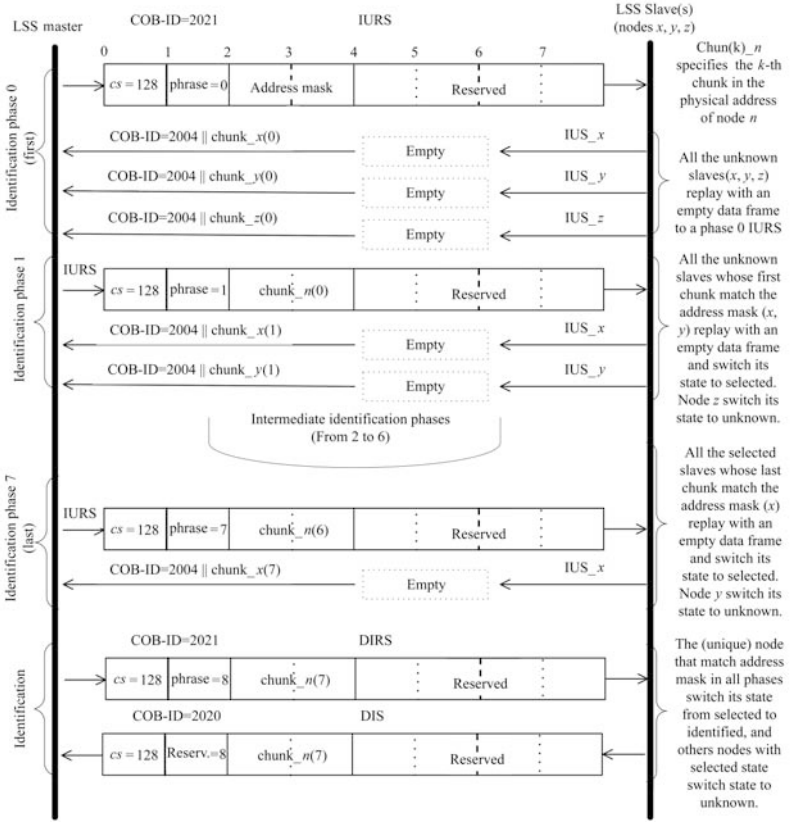

Fig. 10 Frame formats and node discovery procedure

5.2 Performance evaluation

We evaluate the performance in a network simulation tool to confirm the efficiency of node management protocol under practical conditions.

The CANoe tool[27], a network analyzer for CAN systems, is used to trace the frame flows over the bus, and real time messages generated by the virtual nodes is recorded in log files, which reflects the current states and results of the virtual system. By this tool, the correctness and the performance of the binding protocol could be verified. In our scheme, the extended 29 bits frame format of CAN is adopted. The data field of node guarding frame is empty and the response frame is 1 byte. Considering the worst case transmission time on the CAN bus, the length of the node guarding frame is 150 bits and the response frame is 160 bits. A screen-shot of the output obtained by the CANoe is present in Fig. 11. The "trace window" shows the sequence of the frames exchanged over the bus, while the "write window" shows the state of the master and slave nodes as the algorithm proceeds.

From the simulation logs, the time taken to detect 30 online nodes in a 50 kb/s CANopen network (which is very slow) where the worst-case slave response time is 4 ms and the master processing time is equal to 1 ms is at least 190 ms. The assumed conditions will be applied to the following evaluation phase. The time needed to discover 30 unknown nodes is at least 11.2 s, which is shorter than that of the standard LSS service of CANopen (16.09 s). The time cost of binding phase for all these 30 nodes is 1.15 s, and the average is 0.038 s for each known nodes.

6 Conclusions

The dramatic growth of networked control systems confronts designers with serious difficulties of distributed infrastructure complexity, communication environment heterogeneity, and control strategy incompatibility. It has motivated the call for a reconfigurable protocol stack with self-recovery, self-tuning, and self-management. The principles of self-organization can be useful for these goals of autonomous behavior.

In this paper, the contribution is to give the architecture of the reconfigurable protocol stack for a networked control system, propose a triple-layer self-organization structure, and point out its foundation role of implementing the self-organization to deal with the dynamic reconfiguration process. It goes beyond the previous work by extracting common features, which contributes to the self-organized protocol design, as fundamental elements: local interactions and implicit coordination. These two basic design elements suggest a general guideline on the internal and external information exchange in the self-reconfiguration level adaptation. Furthermore, considering one of the fieldbus control system, the dynamic reconfiguration schemes on improving the address management is implemented and verified under the idea of self-organizing reconfigurable protocol stack. It is designed to provide the plug & play mechanism in the existing CANopen architecture.

In the future, under the proposed architecture and management of the reconfigurable protocol stack (see Fig. 4), the detailed self-organization manager specifications and their reference implementations, which are not the focus of this paper, still need researchers′ continuing efforts. This paper focuses on the illustration of dynamic reconfiguration procedure but not on how to evaluate the reconfiguration result before implementation. Thus, one of the next directions is the implementation of management framework based on formal technologies, including the related functionality validation and performance evaluation methods on the formal protocol stack model.

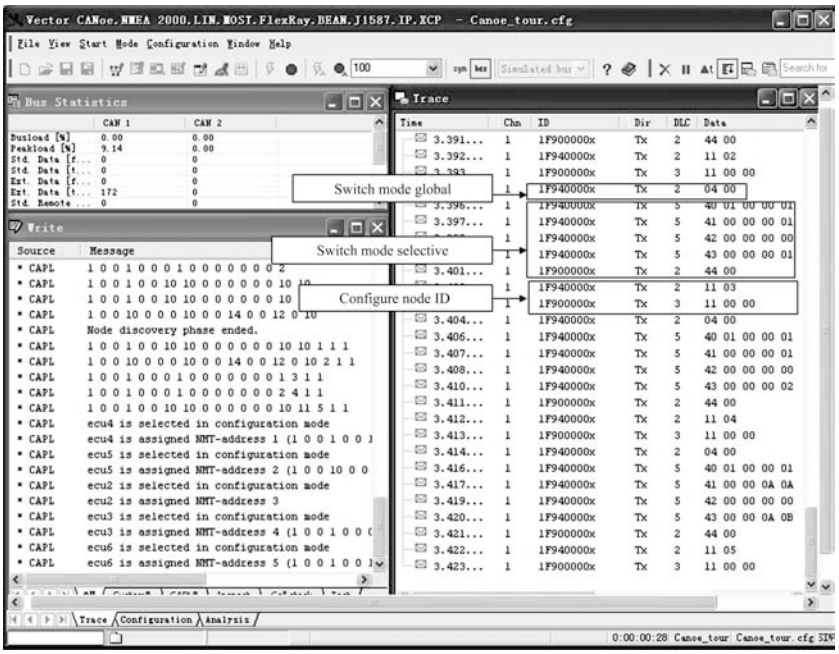

Fig. 11 Sample output produced by the emulation tool

References

[1] S. Kolla, D. Border, E. Mayer. Fieldbus networks for control system implementations. In *Proceedings of Electrical Insulation Conference and Electrical Manufacturing and Coil Winding Technology Conference*, IEEE, Indianapolis, USA, pp. 493–498, 2003.

[2] G. R. Wang, J. M. Qian. CAN bus and the higher layer protocol based on CAN protocol. *Computer Measurement & Control*, vol. 11, no. 5, pp. 391–394, 2003. (in Chinese)

[3] X. M. Tang, J. S. Yu. Feedback scheduling of model-based networked control systems with flexible workload. *International Journal of Automation and Computing*, vol. 5, no. 4, pp. 389–394, 2008

[4] R. Duan, X. Y. Fan, D. Y. Gao, G. Shen. Reconfigurable computing technology and developing trends. *Application Research of Computers*, vol. 21, no. 8, pp. 14–17, 2004. (in Chinese)

[5] O. Lysne, T. M. Pinkston, J. Duato. A methodology for developing deadlock-free dynamic network reconfiguration processes. *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 5, pp. 428–443, 2005.

[6] L. M. An, H. K. Pung, L. F. Zhou. Design and implementation of a dynamic protocol framework. In *Proceedings of the 12th IEEE International Conference on Networks*, Singapore, vol. 2, pp. 552–558, 2004.

[7] M. Muhugusa, G. D. Marzo, C. Tschudin, S. Eduardo, H. Jürgen. Comscript: An environment for the implementation of protocol stacks and their dynamic reconfiguration. *International Symposium on Applied Corporate Computing*, 1994.

[8] W. K. Chen, M. A. Hiltunen, R. D. Schlichting. Constructing adaptive software in distributed systems. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, IEEE, Mesa, USA, pp. 635–643, 2001.

[9] G. T. Wong, M. A. Hiltunen, R. D. Schlichting. A configurable and extensible transport protocol. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, IEEE, AK, USA, vol. 1, pp. 319–328, 2001.

[10] K. T. Seng, G. Yu, S. T. Kean, W. A. Chee, G. Nirmalya. Dynamically loadable protocol stacks — A message parser-generator implementation. *IEEE Internet Computing*, vol. 8, no. 2, pp. 19–25, 2004.

[11] C. Prehofer, C. Bettstetter. Self-organization in communication networks: Principles and design. *IEEE Communications Magazine*, vol. 43, no. 7, pp. 78–85, 2005.

[12] W. Ye, J. Heidemann, D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[13] B. J. Chen, K. Jamieson, H. Balakrishnan, R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

[14] H. Abusaimeh, S. H. Yang. Dynamic cluster head for lifetime efficiency in WSN. *International Journal of Automation and Computing*, vol. 6, no. 1, pp. 48–54, 2009.

[15] D. Senthilkumar, A. Krishnan. Nonsaturation throughput enhancement of IEEE 802.11b distributed coordination function for heterogeneous traffic under noisy environment. *International Journal of Automation and Computing*, vol. 7, no. 1, pp. 95–104, 2010.

[16] L. Yang, X. P. Guan, C. N. Long, X. Y. Luo. Feedback stabilization over wireless network using adaptive coded modulation. *International Journal of Automation and Computing*, vol. 5, no. 4, pp. 381–388, 2008.

[17] J. P. Thomesse. Fieldbus technology in industrial automation. *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1073–1101, 2005.

[18] N. C. Hutchinson, L. L. Peterson. The *X*-kernel: An architecture for implementing network protocols. *IEEE Transactions on Software Engineering*, vol. 17, no. 1, pp. 64–76, 1991.

[19] S. Mena, X. Cuvellier, C. Gregoire, A. Schiper. Appia vs. Cactus: Comparing protocol composition frameworks. In *Proceedings of the 22nd International Symposium on Reliable Distributed Systems*, IEEE, Florence, Italy, pp. 189–198, 2003.

[20] M. Kannan, E. Komp, G. Minden. Design and Implementation of Composite Protocols, Technical Report, Department of Electrical Engineering and Computer Science, Anna University, Chennai, India, 1997.

[21] N. Vatanski, J. P. Georges, C. Aubrun, E. Rondeau, S. L. J. Jounela. Networked control with delay measurement and estimation. *Control Engineering Practice*, vol. 17, no. 2, pp. 231–244, 2009.

[22] F. L. Lian, J. Moyne, D. Tilbury. Network design consideration for distributed control systems. *IEEE Transactions on Control Systems Technology*, vol. 10, no. 2, pp. 297–307, 2002.

[23] D. S. Kim, D. H. Choi, P. Mohapatra. Real-time scheduling method for networked discrete control systems. *Control Engineering Practice*, vol. 17, no. 5, pp. 564–570, 2009.

[24] F. Cen, T. Xing, K. Wu. Real-time performance evaluation of line topology switched Ethernet. *International Journal of Automation and Computing*, vol. 5, no. 4, pp. 376–380, 2008.

[25] C. J. Zhou, C. J. Xiang, H. Chen, H. J. Fang. Genetic algorithm-based dynamic reconfiguration for networked control system. *Neural Computing and Applications*, vol. 17, no. 2, pp. 153–160, 2008.

[26] G. Cena, A. Valenzano. A protocol for automatic node discovery in CANopen networks. *IEEE Transactions on Industrial Electronics*, vol. 50, no. 3, pp. 419–430, 2003.

[27] Vector-Informatik GmbH. CANalyzer User Manual, Version 7.1, [Online], Available: http://www.vector-worldwide.com, December 17, 2010.

Chun-Jie Zhou received the M. Sc. and Ph. D. degrees in control theory and control engineering from Huazhong University of Science and Technology, Wuhan, PRC in 1991 and 2001, respectively. He is currently a professor in the Department of Control Science and Engineering at Huazhong University of Science and Technology.

His research interests include industrial communication, artificial intelligence, and theory and application of networked control systems.

E-mail: cjiezhou@mail.hust.edu.cn (Corresponding author)

Hui Chen received the B. Sc. degree in electrical and electronics engineering from the University of Fuzhou, Fuzhou, PRC in 2005, and the M. Sc. degree in control theory and control engineering from Huazhong University of Science and Technology, Wuhan, PRC in 2007. He is currently a Ph. D. candidate in control science and engineering at Huazhong University of Science and Technology.

His research interests include formal description technology and industrial communication.

E-mail: husthuichen@gmail.com

Yuan-Qing Qin received the M. Sc. and Ph. D. degrees in control theory and control engineering from Huazhong University of Science and Technology, Wuhan, PRC in 2003 and 2007, respectively. He is currently a lecturer in the Department of Control Science and Engineering, Huazhong University of Science and Technology.

His research interests include networked control systems, artificial intelligence, and machine vision.

E-mail: yuan_qing@163.com

Yu-Feng Shi received the B. Sc. degree in the Department of Control Science and Engineering from Huazhong University of Science and Technology, Wuhan, PRC in 2006, and M. Sc. degree in control theory and control engineering from Huazhong University of Science and Technology in 2008. He is currently a Ph. D. candidate in control science and engineering at Huazhong University of Science and Technology.

His research interests include embedded operating systems and industrial communications.

E-mail: syfhust@gmail.com

Guang-Can Yu received the B. Sc. degree in computer engineering from Logistical Engineering University, Chongqing, PRC in 1997, and the Ph. D. degree in computer science and technology from Huazhong University of Science and Technology, Wuhan, PRC in 2008. Currently, he is engaging in postdoctoral work in the Department of Control Science and Engineering at Huazhong University of Science and Technology.

His research interests include industrial communication and software architecture.

E-mail: ygcan@qq.com