Abstract—In NCS (networked control systems), the control performance depends on not only the control algorithm but also the communication protocol stack. This paper presents a framework for the protocol reconfiguration in NCSs. Task queuing, network transmission policy and communication link control are emphasized as three key protocol functionalities related to the system performance. Based on it, data and event exchange models for layers of protocol stack are developed according to the standard IEC61499, which allows for the dynamic reconfiguration management of distributed resources as well as provides a unified real time communication service for control tasks. Finally, experiments are conducted to illustrate the application and efficiency of the proposed reconfigurable protocol stack.

Keywords Architecture; protocol stack; reconfiguration; function block; IEC61499

I. INTRODUCTION

A networked control system uses a distributed control architecture where sensors, actuators and controllers are interconnected through a real time network [1]. It is inherently difficult to guarantee punctuality and predictable quality of communication service (QoS), because the processing overhead of protocol stack is highly influenced by the implementations of software architecture, and QoS objects of different protocols are hard to be coordinated when lacking of a unified architecture that specifies functional and interactions models therein.

To cope with this challenge, there has been an increasing emphasis on developing reconfigurable protocol stacks with the self-adaptability in such a distributed, heterogeneous and changeable environment [2]. Current software engineering such as model driven development may facilitate the development and deployment of complex protocol reconfiguration process [3]. In this concept, the standard IEC61499 provided the means to add or remove the simulation capabilities dynamically to the real device. Thus, the reconfiguration could be automatic i.e., without the intervention of a human administrator and dynamic i.e., without service interruption, or with a minimal one [4]. The protocol design of IEC61499 is still an open issue due to the missing of definitions of real time and reconfiguration executions on the standard communication services.

In this paper, we develop an IEC61499 based framework for the management of reconfiguration process of protocol stack that can resolve heterogeneous and real time problem in NCS. Under the environment of FBDAK [5], we extend the service interface function block (SIFB) of IEC61499 with a layered reconfigurable protocol stack, which allows for the dynamic reconfiguration management of distributed resources and provides a unified real time communication service for the control tasks. It enables a flexible resource allocation scheme at system level by integrating the FBs of application, network transmission and communication link layers. Comparing with the proposed SDL architecture for managing the reconfiguration process [6], this protocol SIFB facilitates the development of IEC61499 applications.

II. IMPLEMENTATION OF THE RECONFIGURABLE PROTOCOL STACK WITH IEC 61499

A. Architecture of Reconfigurable Protocol Stack

In this work, the protocol stack is considered as a special resource model to describe the event and data flow of sharing the channel, memory and computation resources of NCS when a specific communication service is demanded. The resource model specifies the place where an application will be executed. The architecture for internal information exchange inside the protocol stack is shown in Fig.1. Function blocks inside this resource model manage communication activities within interconnected devices and interfaces are provided to the reconfiguration communication that adapts to external environment changes. The reconfiguration needs a cooperative effort of three layer blocks: the communication link layer (CLL), the network transmission layer (NTL) and the application layer (APP).

The APP block is the interface between the protocol stack and users to collect the node information and interpret the user task. It provides high reliable data services for the applications on the User Layer. The interface P_Task and N_P_Task is responsible for input of task queues with periodic and non-periodic. The change of system structure or task requirements would transfer through the data channels of “Node_Info <-> TaskTable” and “Sys_Info <-> TopoStructure”, which reinitialize the configuration of platform resources that are required by the scheduling and routing algorithm of NTL.
Figure 1. The resource model of reconfigurable protocol stack.

The NTL block is mainly composed of three parts: the data channel, the scheduling scheme and the routing scheme. The scheduling table is constructed as a specific string format - SchMSG, which is broadcast to the subscribed devices to control the traffic between the APP and NTL. The routing scheme determines the NextHop address for data forwarding with the assumption that transmission delays are all bounded into a certain range.

The CLL mainly represents the functionalities of Physical Layer (PHY) and the Data Link Layer (DLL). The port QoS Change is to report the peer to peer link status. Once an error happening, it would activate the request for routing reconfiguration (RouteConf_REQ), and the accomplishment of re-routing process (RouteCNF) will require a period of configuration mode of communication controller chip for updating the information of NextHop address (NextHopAddr).

B. Protocol Stack Components

1) Application layer

As depicted in Fig.2, the initial state of START is to indicate that the protocol stack is ready for providing communication service.

a) INIT: It is the initialization state to prepare the information of task table and topology structure for reconfiguration activities.

b) Proc_NonPeriodic & Proc_Periodic: They are the states represent the queuing of non-periodic and periodic tasks. The algorithms Q_POP and Q_PUSH are the behaviors of adding and removing a task from the task queue in a FIFO style. When scheduling reconfiguration occurs in NTL, the algorithm Q_PUSH is activated to sort the sequence of task queues in accordance with the scheduling table. The inputs for the Q_PUSH are different in the Proc_NonPeriodic and Proc_Periodic, that is, the former is performed according to trigger of SchMSG (the re-scheduling result) and the latter will run immediately when a non-periodic task declaration of high priority is received.

Figure 2. The ECC of APP block.
c) SEND & RECEIVE: They are the states represent the process of coding and decoding the messages for user tasks. In the case of PLC (Programmable Logic Controller) applications the data contents and format through the I_Data and O_Data port are varied with different command types.

2) Network transmission layer
The NTL is to provide a QoS guaranteed scheme for the stabilization of environments variation and optimization of control performance (Fig. 3).

a) TransControl: It is the state that is under the control of scheduling signals. It comprises two processes: one is for real time traffic – real time (RT) channel, the other is for non-real time traffic – non-real time (NRT) channel. If transmission time expires, the outdated message would be discarded and the application layer would be informed that there is no time left for the execution of message retransmission.

b) ReRouting: This state is the kernel of routing reconfiguration, which is followed by the missing or failure result in the state PathQuery and determines the address of NextHop for the state Forwarding. The received message will be forwarded to the CLL directly by unicast on the case that the destination is in the same network segment; otherwise, it will be conveyed to the neighboring segment in a multi-cast style. Fig. 4 shows the workflow of the reconfiguration of routing table. Once a transmission request appears on the NTL, the routing frame is formed with a general data structure as “Preamble | SFD | Destination Address | Source Address | Protocol Type | Data | Padding | FCS”. All the referring segments should follow a certain communication standard, such as ISO11898 for CAN, IEEE 802.3 for Ethernet, and IEEE 802.15.4 for Zigbee. And, depended on different QoS levels, the “Data” field could be varied from different routing service. For instance, the command ID could be neighbor discovering, address binding, node add-in and so on.
c) ReScheduling: This state is activated when task table changes. The scheduling message (SchMSG) is generated in the states of synchronous transmission (SynTransmission) and asynchronous polling (AsynPolling). The data structure for task table, scheduling table and the SchMSG are shown in Fig.5. Firstly, the operation of scheduler works in a style of plan to plan, i.e., replaces the old scheduling table (i) with table (i+1) to cope with possible changes in the task set and the variation of QoS. The algorithm in the state ReScheduling could be reconfigured with the component of weight fair queuing (WFQ), early deadline first (EDF), or rate monotonic (RM). Secondly, the scheduling table is formed as a set of element cycles (EC) that represent the traffic within a certain period. The size of each scheduling table should be equal to a Macro Cycle (MC), which comprises enough ECs for polling all the tasks of slaves once at least in a MC. Each EC is filled in the data field and formatted as “Net_ID | Number of Tasks | (Task ID, Len) | … | (Task ID, Len)”, where number, ID and length of task represent the total traffic within one network segment and time slice for each task.

Figure 5. Data interface for the scheduling reconfiguration.

3) Communication link layer
The CLL is to provide a reliable transmission link (Fig.6). The ECC shows the switching between the mode of normal operation and fault processing for hardware configuration.

b) Fault: It is the state for the fault processing that could be activated by the confirmation of routing reconfiguration. The fault may be of two types, one is referred to the link error, and the other is caused by the conflict. For the former, the fault processing in this block will prevent the next message entering the SEND state. The latter requires a period of backoff time for the release of communication resources.

c) TimeSynchronous: It is an optional state when users employ a TDMA based medium access mechanism. It requires a timer in all nodes of network for the support of a precise time triggered scheduling scheme.

III. PERFORMANCE EVALUATION
A. Test Bed
The IEC61499 standard defines an independent, model-based, reused, and distributed method for industrial process and measurement control system. As shown in Fig.7, the proposed
test bed is composed of four modules: the Master Manager, the Communication Device, the PLCs, and the I/O type of actuators. It is a layered network topology with hybrid communication services: LocalBus, Ethernet, DeviceNet. These three protocol segment model is heterogeneous internally, thus, the Communication Device module provides the channel between the Master Manager and the Ethernet segment. It is a special device that contains the developed protocol stack resource with the reconfiguration ability to cope with various and changeable traffic types in the Ethernet segment. Supported by IEC61499, such a protocol stack resource model could be dynamically migrated onto any position (device model) of the system by specifying the reconfiguration service of local or remote device management models.

B. Experiment Results

We deploy a test on the time cost for the reconfiguration of CLL layer block, which refers to the instantiation of an Ethernet-based CSMA algorithm in the CLL block by a set of creating and deleting operations. Therefore, the time cost for reconfiguration can be calculated following function:

\[ T_{\text{reconfiguration}} = T_{\text{generation}} + T_{\text{distribution}} + T_{\text{loading}} \]

where it consists of three parts: the code generation time, the code distribution time, and the code loading time. The result is platform dependent and affected by various issues such as size of code file, efficiency of compiler, congestion status of network and processor capability.

The accuracy of system clock in the measurement is 10 ms. The time cost for code generation, distribution and loading are summarized in Table I. The execution time of code generation and loading process is notable 96% of the total execution time for reconfiguration, the former takes 25s on the PC for preparing the new code, and the latter takes 5s on the PLCs for launching a new communication protocol stack. These two processes are executed offline (i.e., computed in an independent computing unit) without having impact on communication activities of the old protocol stack during the reconfiguration execution period. The distribution of the new code is scheduled as the non-periodic non-real time traffic with assigning a fixed time slice in each macro scheduling cycle. In this way, its impact on the real-time traffic could be controlled at a low level. Thus, though the transmission delay is very small, the average value for the distribution time could reach 1295ms, which is caused by the postponed sending in each scheduling cycle.

TABLE I. TIME COST FOR THE RECONFIGURATION OF CLL BLOCK

<table>
<thead>
<tr>
<th>T</th>
<th>Generation (ms)</th>
<th>Distribution (ms)</th>
<th>Loading (ms)</th>
<th>Total (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>22960</td>
<td>1220</td>
<td>4850</td>
<td>29030</td>
</tr>
<tr>
<td>M</td>
<td>28280</td>
<td>1340</td>
<td>5420</td>
<td>35040</td>
</tr>
<tr>
<td>M</td>
<td>25190</td>
<td>1295</td>
<td>5200</td>
<td>31685</td>
</tr>
</tbody>
</table>

(T: Time Cost, M: Measurement)

IV. CONCLUSION

The dramatic growth of networked control system confronts designers with serious difficulties of distributed infrastructure complexity and communication environment heterogeneity. The contribution of this paper is to propose an IEC61499 compliant feedback structure for the reconfigurable protocol stack of NCS. The integrated resource model (in Fig.1) suggests a general guideline on the protocol layer interactions. It can serve as an extension communication block to the standard IEC61499. In future work, deeper investigations on the protocol service models are needed for the performance testing, especially the ad hoc environment where the complexity of channel models should be concerned.

REFERENCES