A Protocol for Automatic Node-ID Binding in CANopen Networks

Guangcan Yu
Department of Control Science & Engineering, Huazhong University of Science and Technology, Wuhan, China
Email: ygcan@QQ.com

Chunjie Zhou and Shuang Huang
Department of Control Science & Engineering, Huazhong University of Science and Technology, Wuhan, China
Email: cjiezhou@mail.hust.edu.cn; huangshuang0709@126.com

***Abstract*—In a CANopen-compliant industrial control system, the master node controls slave nodes (devices) through the corresponding Node-IDs. Though there are some protocols (such as LSS, LMT etc) established for Node-ID assignment, it is not competent for configuring Node-ID for all slaves automatically. In this paper, a new Node_ID binding protocol is proposed. The protocol can be used to assign Node-IDs to all the slave nodes of a networked control system automatically, and the assigned Node-ID can also reflect the logic position of a physical slave node in the control system. Firstly, the LSS physical address structure is applied to a developed protocol for automatic node discovery (AND) with some necessary modifications. The revision of the AND protocol (RAND) is proved to be able to detect LSS addresses of all slaves efficiently. After all the slave nodes being discovered, each slave node needs to be assigned with an appropriate Node_ID. In the proposed protocol, the assignment is divided into two phases: the Node-IDs pre-assigning phase and the Discovery and binding phase. The accomplished function, the implementation rules and the sorting algorithms of the two phases are described in detail. Finally, the performance of the protocol is evaluated both from a theoretical point of view and by means of a network emulator. The latter approach also verified the correctness of the protocol under actual conditions. It is illustrated that with the proposed protocol, it is easier to deploy and maintain a control system.**

***Index Terms*—Communication protocols, CANopen, Industrial control system, Node discovery and binding**

I. INTRODUCTION

CAN[1] is a real-time protocol that is widely adopted in a number of different application fields[2, 3] , ranging from the automotive to the automated manufacturing environments. The CAN specifications define only the data-link layer, which describes the rules used for carrying out frame exchanges over the shared communication support. A higher level protocol is necessary to specify the process of CAN message frames. CANopen[4] is an application protocol conceived for frame exchanges in distributed control systems based on CAN concepts. CANopen was originally proposed in the framework of a European research project and is now maintained by the CAN-in-Automation (CiA) organization. Recently, it has become a European standard known as EN 50325-4[5]. A CANopen system is typically comprised of one master and several slave devices (slave nodes), in which each slave node is identified by a 7-bit node address (Node-ID) and message exchange is achieved by communication objects (COBs). There are mainly two kinds of COBs: process data objects (PDOs) and service data objects (SDOs). The former is used to transfer real-time data from one producer to (one or more) consumers; the latter is subjected to node configuration and managements, and provides a client access to entities defined in a device object dictionary (OD). However, Node_IDs and the bit timing parameters could not be configured by SDO because communication with a given device is not possible unless it is assigned with a unique node ID and the correct bit rate has been set. The method of the bit timing configuration is outside the scope of the paper, we focus on how the Node_ID can be configured.

The Node-ID can be configured either manually (by means of dip-switches), or via the CAN bus. If slave devices were provided in sealed packages (for the purpose of to be resistant to water, dust, sand or other hostile agents), the latter is often the only practical method. Two protocols which support the Layer Management (LMT) and Layer Setting Services (LSS) respectively can be used to configure the Node-IDs. LMT[5] is an earlier solution, features in its ability of changing values of the Node_IDs. However, the whole system could break down if two or more slave nodes were tagged with the same Node_IDs. The latest version of the CANopen specifications is based directly on the communication facilities as set in the CAN data-link layer, leading to the introduction of LSS[6], which is quite similar to the LMT but was essentially tailored for the CANopen architecture. It requires that there must be only one node in the configuration mode while others in the operational mode if LSS is utilized as the

Manuscript received April 13, 2012; revised July 8, 2012; accepted July 31, 2012.

This study was supported by National Natural Science Foundation of China (No. 60674081 and No.61074145), Independent Innovation Research Foundation of Huazhong University of Science and Technology, China(No. 2011TS029).

doi:10.4304/jcm.7.10.765-773

configuration tool. Tindel[7, 8] present a static approach for assigning identifiers to different CAN messages according to the deadline monotonic scheme. The CAN application layer, instead, defines the distributor (DBT) protocol[9, 10], which enables the dynamic distribution of the identifiers through a suitable COB database. Finally, Cavalieri[11] introduced a novel protocol based on DBT and copes with the online automatic management of the identifiers. Although these protocols could meet vary requirements, they are not competent for Node-ID binding automatically. When these protocols are used to configure Node_ID for slave nodes, the following methods would usually be adopted. (ⅰ) Before a slave node is connected to a control system, the Node_ID is configured by a special auxiliary system (tool) manually, and the operator must ensure the correctness and uniqueness of the assigned Node_ID. (ⅱ) If slave nodes possess globally unique identifier (such as LSS address), a static mapping table which maps each slave node to a Node_ID should be provided to every deployed control system. When any slave node is replaced, the static mapping table needs modified manually. In general, the two methods are unacceptable in industrial fields.

In order to automatically configure Node-IDs for all slave nodes in a CANopen network, each slave nodes need an inherent identifier which may be global unique and may not be modified. The LSS address is suited to act as the inherent identifier. As the dynamic host configuration protocol (DHCP)[12], the MAC addresses are utilized to configure IP address. At the same time, a discovery mechanism is needed to detect all the inherent identifiers automatically. Although LSS identification services is theoretically possible in discovering unknown nodes that have already been connected to the network, it would require an exhaustive scanning of all the 16-byte LSS addresses, and that is too difficult in real implantations. Cena and Valenzano[13] presented an efficient solution for rapid recognition, the Automatic Node Discovery (AND) protocol, which does not need any priory information about the unknown nodes. Such a technique is based on the fact that each device can be identified unambiguously by a global unique identifier. It adopts a node identification procedure which comprises a sequence of identification phases. 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 respond and, thus, it is identified. Such procedures will be repeated until all nodes become known in the network. The AND protocol is based on four new kinds of unconfirmed services: identify unknown remote slaves (IURS), identify unknown slave (IUS), direct identify remote slave (DIRS) and direct identify slave (DIS). These four services form the node identification procedure. 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). For this purpose, two additional switch identification state services (global and selective) are provided. The application of AND protocol is limited because of its 6-byte physical address does not conform to the LSS specifications, and too short to express certain semantic information.

The proposed automatic Node-ID binding method aims at configuring Node-IDs for all slave nodes in a CANopen network automatically and the configured Node-IDs may also reflect the logic position of a physical slave node in the network. The LSS address is adapted to global identify each slave node and is utilized to configure Node-ID for each slave node. Our method is based on the following assumption: In a CANopen-compliant control system, the CANopen network is always static, the number of slave nodes and the functionality of each slave node are usually pre-determined. Slave nodes can be classified by the type codes (comprised of vendor-id and product-code) in LSS address, and slave nodes with the same type code may be equal in functionality. And then, we divide the Node-IDs assigning work into two phases. In the first step, every Node-ID is pre-assigned to a slave node according to the type code, and the mapping relation between Node-IDs and type codes is stored in the master node. The Node-IDs are agents of the physical slave nodes and the master node would control every slave node though the corresponding Node-ID. In the second phase, slave nodes would be connected to the network according a given rules, and the Node-IDs can be automatically assigned to the corresponding slave nodes which are connected to the network according to the Node-ID pre-assigning. Slave nodes store the assigned Node-IDs, and prepare for responding messages with given Node-IDs. The Node-ID assignment work is made up of two phases, and only in the second phase the actual work is done automatically. In fact, it is very important, for it is not good ideal to expect workers of industrial fields to do some complicate configure work, and our method makes it easier to deploy and maintain the control system. If any slave node failed 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.

Our automatic Node-ID binding method is based on the LSS addresses of slave nodes. Before Node-IDs can be assigned to slave nodes, the master node needs to know LSS addresses of all connected slave nodes. A revision of the AND protocol (RAND) is proposed and can be used to detect all slave nodes. In RAND, the LSS address is adopted instead of the 6 bytes self-defined address, and the extended 29-bit frame format of CAN is adopted instead of the standard 11-bit frame format, and some necessary corresponding modifications are made. The RAND is proved to be able to detect LSS address of all slaves efficiently.

This paper is structured as follows. Section 2 presents the revision of the AND protocol. Section 3 presents the new Node-ID binding protocol. In Section 4, the correctness and the performance of the proposed

improvements will be evaluated. And finally, a summary will be made in Section 5.

II. Revision Of The And Protocol

In RAND, The LSS address, which is significantly longer than the 6-byte physical address in the AND protocol, is split up into eight chunks as depicted in Fig. 1. Here, the extended 29-bit frame format of CAN is adopted instead of the standard 11-bit frame format, for fully utilization of the address space, and thus better performance could be expected.

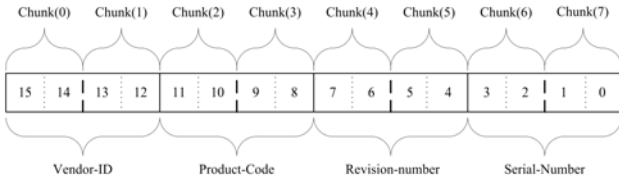

Figure 1. LSS address structure.

The IURS frame uses the COB-ID 2021. It contains a 4-byte data field, and the command specifier (cs) 80H is included in the first byte. The second byte stands for the phase number, its binary value ranging from 0 to 7. The successive two bytes are the address mask, and the remaining 4 bytes are reserved. The phase number is corresponding to the address mask. (If the phase number is 0, the address mask is set to null; if the phase number is 1, then the mask is set to chunk (0) of LSS address, and so on).

The IUS frame has an empty data field. The first 11 bits of COD-ID is 11111010100b (2004), and the following 16 bit is to be filled with a chunk of LSS address. The last two bits are reserved.

The DIRS frame format is similar to IURS, the phase field is set to 8 and the last chunk of LSS address is specified as the address mask. The DIS frame is similar to IUS too, but its COB-ID is 2020.

Because the CAN frame has only an 8-byte length data field, the 16-byte LSS address is obvious too long to be filled in the data field together. Thus the meaning of the status flag (identification state) as defined in AND protocol, which is stored in each slave node and indicates whether the node itself is identified by the master node or not, need to be extend to express two statuses. The new status is named selected, which indicates that a node has accomplished all prior identification phases in a single

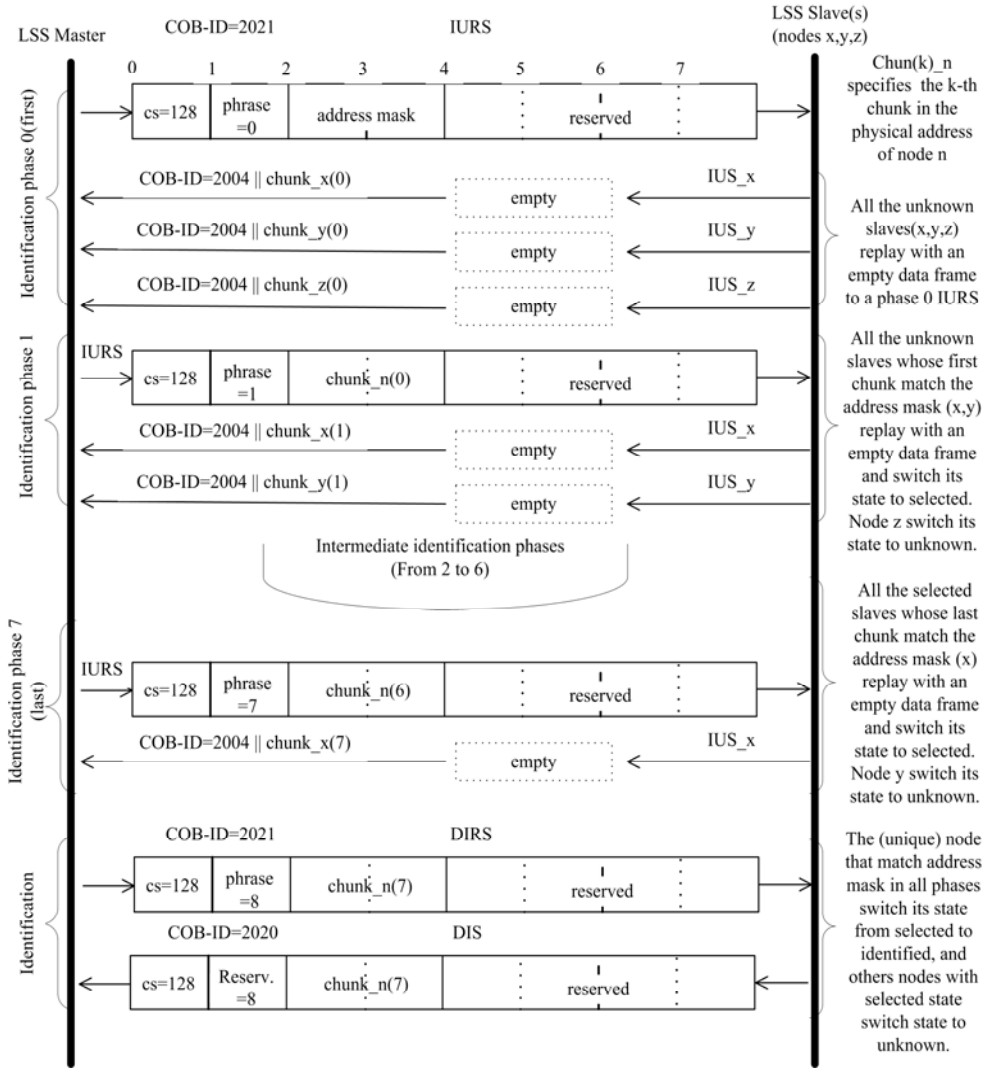

Figure 2. LSS address structure. Frame formats and the protocol for identifying one unknown slave device

procedure, and is ready to respond to the next phase.

If a slave has received a zero phase IURS frame and in the unknown state, or an none zero phase IURS while in the selected status, then the corresponding part of its physical address will be checked to determine whether it matches the address mask specified in the IURS frame. If so, the node will respond with a suitable frame and its status will be switched to selected. If not, the status will be switched to unknown and no actions will be taken. For example, if the current phase number is zero, then all the unidentified nodes have to respond. If the phase number equal to 6, it will involves only those nodes, whose chunk (5) of the physical address are identical to the 16 most significant bits in the address mask of the IURS frame, and whose status is selected. Those nodes with selected status, whose chunk (5) of the physical address are not identical to the 16 most significant bits, would switch their status from selected to unknown.

Data fields of all the responding frames are empty, as it is detailed in the AND protocol. But the COB-ID assigned to the service is different from the AND protocol specifications. In fact, COB-IDs are calculated by attaching two bytes to the value 2004. The two bytes are the chunk in the physical address that corresponds to the current phase number (for example, in phase 0 the chunk (0) is selected). We define the symbol "||" as the operation of a chunk attaching to another number (for example, if the corresponding chunk value is 2, the COB-ID= 2004 (11bit) || 2 (16bit) || 0 (2 bit reserved) = 11111010100000000000000001000b, the last two reserved bits are omitted in the paper for simplicity). The format of the frames used to support the services and the protocol for identifying one unknown slave device are shown in Fig. 2.

III. The Automatic Node-ID Binding Protocol

CANopen network usually adopts the Client/Server model, in which the master node serves as the server and others, clients. The server node can be the NMT master or LSS master, etc. In a CANopen-compliant control system, the number of slave nodes and the functionality of each slave node are usually determined, and the CANopen network is static. A general architecture of CANopen networks is shown in Fig. 3.

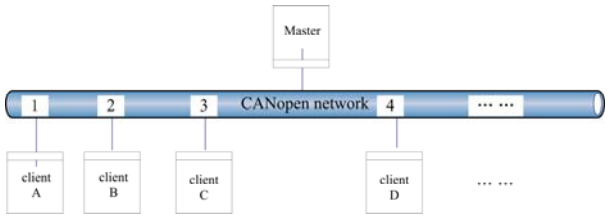

Figure 3. General architecture of CANopen networks.

Each slave node is connected to the network through a numbered slot. According protocol specifications, the maximal amount of nodes connected simultaneously to a CAN bus is 127. The slots could be numbered from 1 to 127 (they can also be given a semantic name). The automatic Node-ID binding protocol aims at assigning each slot number to the slave node which is connected to the corresponding slot and the assigned slot number will be used as Node-ID of the slave node. So the Node-ID also reflects the logic position of a physical slave node in the network. For example, the slot number 1 is assigned to client A (that is, the Node-ID of the client A will be 1). It's natural that the assigning work could be done manually: record all LSS addresses of slave nodes and the slot number of each node, and then the assigning work could be achieved through certain HUI of the master node. However, if this process needs to be accomplished without human interference, the most challenging task is to determine the corresponding slot number of each slave node automatically.

Automatic Node-ID Binding algorithm is actual a sorting algorithm on LSS addresses. As mentioned above, an LSS Address consists of a vendor id, a product code, a revision number and a serial number. Nodes with the same vendor id and product code are equal in functionality, therefore vendor-id and product-code together could be jointed as Type Code. After the functionality of a control system is decided, the node types and the amount of nodes belonging to each type may also be determined. The Node-ID assigning work is divided into two phases in general. In the first phase, slave nodes are not connected to the network yet, and the master node doesn't know LSS addresses of any slave node in detail, but the master node knows the node types and the amount of nodes belonging to each type. In this phase, a node type is mapped to a Node-ID cluster, and the mapping work is done in the master node. We name this phase as Node-IDs pre-assigning phase. In the second phase, slave nodes would be connected to the network first according the following two rules:

Rule 1: The Type Code of a slave node must match Type Code which the slot number (Node-ID) is mapped to, and the slot number represents the slot which the slave node is connected to.

Rule 2: If a node type is mapped to multi Node-ID (Several slave nodes would be equal in functionality and possess the same Type Code), the slave nodes will be sorted by Revision_Number and Serial_Number (R_SN) first. The R_SN value of a node is associated with its slot number: a node with a larger R_SN is connected to a slot with larger number.

After slave nodes are correctly connected to the network, The RAND protocol is used to detect all LSS addresses of all slave nodes. And then, Node-IDs are assigned to slave nodes using the same rule, and the assigned Node-ID of a slave node would be equal with the corresponding slot number and stored in the slave node. Thus the Node-ID Binding is achieved, and the binding work is accomplished though message exchanging between the master node and slave nodes. We name the second phase as discovery and binding phase.

In fact, we use the LSS protocol to assign a Node-ID to a slave node. When LSS is utilized as the configuration tool, it requires that there must be only one node in the configuration mode while others in the operational mode and the following three services are involved:

Switch Mode Global Service is used to switch all LSS Slaves connected to the network between operational mode and configuration mode.

Switch Mode Selective Service is used to switch the LSS Slave, of which LSS address attribute equals to LSS_address specified in the request frames, into configuration mode.

Node-ID Configuring Service is used to configure the NMT-address parameter of a LSS Slave by the LSS Master. This service requires only one LSS Slave in configuration mode.

Then, Configuring a Node-ID involves the following three steps:

(1) Switch all LSS slaves to operational mode with switch global service.

(2) Put selected slave nodes into configuration mode using Switch Mode Selective Service.

(3) Use Node-ID Configuring service to assign the Node-ID to the selected node which is the only slave node in configuration mode.

A. The Integral Sliding-mode Control

In the phase, slave nodes are yet not connected to the network, and the master node doesn't know LSS addresses of slave nodes in detail, but it know the type code of every slave nodes which is part of the LSS address and represent functionality of a slave node. A node type is mapped to a Node-ID cluster, and the mapping relation would determine the Node-IDs range of a type of nodes which are equal in functionality.

For registering all slave nodes and saving the mapping relation between the Node-IDs and the type codes, the master node need maintain a slave node information table (Node_Table). The structure of Node_Table is shown as Table 1. Each row of the table shows the properties of a slave node: the Node-ID corresponds to the slot number that the slave node is connected. The Type_Code is the vendor-id and product_code of a LSS address, and R_SN stands for Revision_Number and Serial_Number of the LSS address. The isOnline indicates whether the slave node has been connected to the network or not and whether it is proper functioning. The value of Node-ID and Type_Code reflect the mapping between node type and Node-IDs cluster, and could be determined after the functionality of a control system is determined. The value of R_SN and is Online could be set only after slave nodes are connected to the network and the automatic binding procedure is successfully executed.

TABLE I.
STRUCTURE OF THE NODE_TABLE

Node_ID	Type_Code	R_SN	isOnline
1	0x**01000001**00000001		
2	0x**00000001**00000001		
3	0x**00000001**00000005		
4	0x**00000001**00000001		
5	0x**00000001**00000001		
…	…		

B. Discovery and Binding Phase

The discovery and binding phase consists of the following three steps in detail:

a) Preparation step

In the step, the master detects all slave nodes through NMT node guarding service one by one. If a slave node is connected and properly functioning, it will response to the node guarding frame initiated by the master, and the isOnline field of a row which represents the slave node is set to true.

If node guarding frame is not responded, both isOnline and R_SN fields are set null, this would state that the slave node is not connected or not normal running.However, if the system is in first time initialization, the R_SN field of Node_Table will be empty because the nodes discovery phase has not yet started. If unknown slave nodes would be connect to a network that has already been configured, or some failed nodes were to be replaced, the cleaning work can be done in the initialization phase.After that, the R_SN and isOnline fields of fault nodes would be reset, with the Node-ID and Type_Code fields remain unchanged. The master node is then ready to accept new nodes after fault nodes are removed from the network by the system manager. The master node can also check on the current states of individual nodes with Node Guarding. A detailed pseudo-C code description of the initialization phase is given in Fig. 4.

```
Node_Table[MAX_Node];
Boolean isOK;
For(int i=0;i<MAX_Node;i++)
      {Node_Guarding(Node_Table[i].Node_ID);}
                        //send node guarding frame
Sleep(T_g_coll); //wait for collecting response frame
For(int i=0;i<MAX_Node;i++){
    isOK=Node_Guarding_Response(Node_Table[i].Node_ID);
                  //search the response frame with
                  //Node_ID as Node_Table[i].Node_ID
     If(isOK) Node_Table[i].isOnline=true
     Else{
          Node_Table[i].isOnline=null;
          Node  Table[i].R  SN=null; }}
```

Figure 4. Pseudo code for algorithm in phase 1.

b) Discovery step

In the discovery step, the RAND protocol as presented above detects any unknown nodes. Before this searching mechanism begins, the following steps need to be executed:

(1) The *LSS switch mode global frame* is broadcasted by the master to switch all slave nodes to configuration mode. The RAND protocol is based on LSS protocol, services defined in the RAND could only be available in configuration mode.

(2) The RAND switch identification state global frame is broadcasted by the master to switch all slave nodes to *unknown* state. State of nodes which are newly connected to the network may be *identified* (nodes come from other running automation systems), and the situation will obstruct the RAND protocol.

(3) The RAND *switch identification state selective frame* is send to each slave node which is registered in Node_Table. The states of slave nodes, which are already registered in the master node, are transfer to *identified*, which will improve efficiency of the RAND.

After these three steps, the RAND protocol can be used to detect the connected but not recognized nodes. A detailed description of the discovery mechanism is given in Fig. 5.

```
Node_Table[MAX_Node];
Discovered_New_Node[MAX_Node];
Int newNodeAmount;
Boolean isOK;
Switch_mode_global(true);
// switch all slave nodes to configuration mode
Switch_identification_state_global(False);
// switch all slave nodes to unknown state
For(int i=0;i<MAX_Node;i++){
            If(Node_Table[i].R_SN!=Null)
//switch a registered slave node to identified state
Switch_identification_state_selective
    (Node_Table[i].Type_Code,
Node_Table[i].R_SN);}
```

Figure 5. Pseudo code for algorithm in phase 2.

c) Binding step

In the binding step, nodes recognized in discovery phase, of which the LSS addresses are stored in array Discovered_New_Node, are then assigned with NMT addresses (Node_IDs). The Node_ID assignment relies on a particular sorting algorithm on LSS addresses.

```
Node_Table[MAX_Node];
Discovered_New_Node[MAX_Node];
// sorted LSS addresses of new discovered nodes (sequence)
Int newNodeAmount;
// amount of new discovered nodes
For (i=0;i<newNodeAmount;i++)
{ For (j=0;j<MAX_Node;j++){
  // Type Code matching
  If (Node_Table[j].Type_Code==
      Discovery_New_Node[i].byte(0..7)
      and Node_Table[j].R_SN==NULL)
          //the j(th) row of Node_Table yet not be registered
    {Node_Table[j].R_SN= Discovery_New_Node[i].byte(8..15)
          //register a slave node
     Node_Table[i].isOnline=true;} } }
For (i=0;i<MAX_Node;i++)
{If(Node_Table[i].R_SN!=Null and Node_Table[i].isOnline==false)
   {Switch_mode_global(false);
    // switch all slave nodes to operation state
    // switch selected nodes to configure state
    Switch_mode_selective
    (Node_Table[i].Node_Type,Node_Table[i].R_SN);
    //assigning the Node_ID to the slave
      node with the specified LSS address
    Configure_Node_ID(Node_Table[i].Node_Type,Node_Table[i].R_SN,
     Node_Table[i].Node_ID); } }
```

Figure 6. Pseudo code for algorithm in phase 3.

After Note_IDs have been pre-assigned to each newly discovered node in the Node_Table, 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 be in configuration mode in a time, so as to conform to the constraints of the LSS *Configure Node-ID* service. A detailed description of this sorting algorithm is given in Fig. 6, and the general time sequences of frame exchanges between the master and the slave nodes is given in Fig. 7.

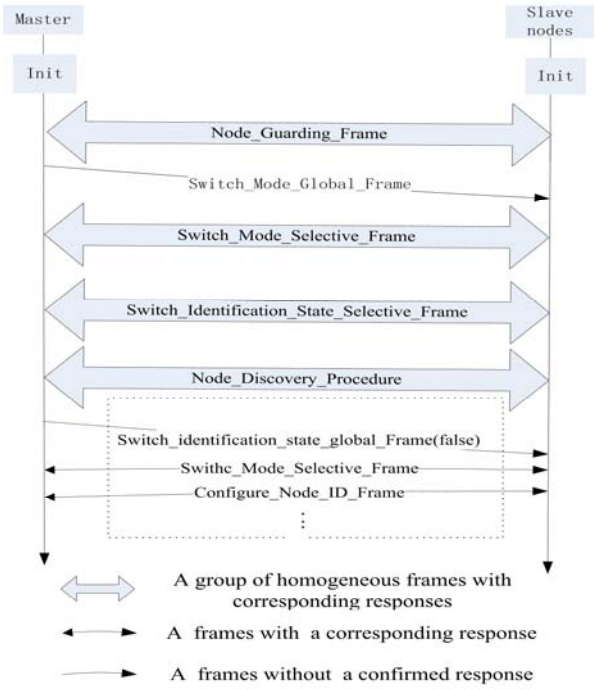

Figure 7. Time sequences in frame exchanges between the master and the slave nodes.

Ⅳ Evaluation Of The Binding Protocol

We evaluate the performance of the proposed protocol from a theoretical point of view first, and then a network emulator is used to confirm the results and the correctness of the protocol under practical conditions.

In the preparation step, the master inquiries each slave node with NMT *node guarding service*. The detection is achieved by the master broadcasting a node guarding frame to the slaves, which are expected to reply with responding frames. The master has to wait for the response for a certain time. Thus, the time needed in discovering one node could be described as the formula given as below,

$$T_{detect} = T_{n\_guard} + T_{n\_guard\_r} + T_{resp} \quad (1)$$

where T_{n_guard} and $T_{n\_guard\_r}$ are the durations of the broadcasting and responding frames respectively, and T_{resp} is the waiting time of the master node (Its value may equal to the maximum value of $T_{n\_guard\_r}$ and we may assume it as a constant). T_{detect} is the time needed for detecting one slave node.

Therefore, the time needed in discovering all slave nodes in the network (T_{detect_all}) could be calculated by

$$T_{detect\_all} = N_{s\_node} \times (T_{n\_guard} + T_{n\_guard\_r}) + T_{resp} \quad (2)$$

where Ns_node is the total number of slave nodes in the system.

Our technique adopts an extended 29-bit CAN frame format. The data field of node guarding frame is empty and the response frame is 1 byte. Consider the worst scenario of transmission on the CAN bus: the length of the node guarding frame is 150 bits and the responding frame is 160bits. Assume the following running conditions (which will also be applied to the following scenarios): Number of online slave nodes is 30, baud rate of the CAN network is 50kb/s (which is very slow), the worst-case slave response time is 4 ms, and the master processing time is 1 ms. Then, the time needed in the preparation phase is 190ms. If the control system is first configured, the value of T_{detect_all} is zero because the R_SN field of the Node_Table is empty (the discovery phase has yet not started).

In the discovery step, the RAND protocol is used to discovery nodes which are newly connected to the network but yet to be recognized. According to the AND (nonrecursive) algorithm, the identification of each unknown node requires 12 identification phases, plus one final direct identification phase. However, in RAND protocol, the count of the identification phases is 8. Similar to the AND protocol, each identification phase of the RAND consists of an IURS frame sent by the master, a time interval , during which replies from the slaves are collected, and some processing time(e.g. during which the master analyzes replies and computes the content of next IURS frame). This could be expressed as below,

$$T_{collcet} \geqslant T_{resp} + (8+1)T_{IUS} \quad (3)$$

$$T_{ID\_phase} = T_{IURS} + T_{collect} + T_{proc} \quad (4)$$

Where T_{IURS}, $T_{collect}$, T_{proc} are the time needed for sending the IURS frame, collecting replies and processing respectively.

The time needed to direct identification(T_{direct_ID}) is

$$T_{direct\_ID} = T_{DIRS} + T_{resp} + T_{DIS} + T_{proc} \quad (5)$$

Where T_{DIRS} and T_{DIS} are the durations of the DIRS and DIS frames respectively.

The time needed to identify one node(T_{node_ID}) is

$$T_{node\_ID} = 8 \times T_{ID\_phase} + T_{direct\_ID} \quad (6)$$

The time needed to discover all the unknown nodes in the network is

$$T_{network\_ID} = N_{unknown} \times T_{ID\_phase} + T_{ID\_phase} \quad (7)$$

Where $N_{unknown}$ is the total number of unknown nodes.

The data field of IURS, DIRS, and DIS frame is 4 bytes, while that of the IUS frame is empty. The identifier of all the frames is 29 bits. In the worst transmission scenario as mentioned above, the lengths of the two frames are 185 and 145 bits respectively. According to (3), the collect time is greater than 30.1ms (taken as 40ms). The time needed to discover 30 unknown nodes is at least 11.2s. Under the same assumed conditions, the time consumed is shorter than that of AND protocol (16.09s).

In the binding step, nodes discovered in the discovery phase are assigned with NMT addresses (Node_ID). To bind a new node involves a *switch mode global frame* (duration T_{s_global}), four *switch mode selective frames* (duration $4T_{s\_sele}$) and a corresponding response frame (duration $T_{s\_sele\_r}$), a configure Node-ID frame (duration T_{c_ID}) and a corresponding response frame (duration $T_{c\_ID\_s}$), and some waiting time consumed by the master (T_{resp}).The total time needed to bind a node (T_{a_bind})is therefore given as:

$$T_{a\_bind} = T_{s\_global} + 4 \times T_{s\_sele} + T_{s\_sele\_r} + T_{c\_ID} + T_{c\_ID\_s} + 2 \times T_{resp} + T_{prcc} \quad (8)$$

Suppose the number of newly discovered nodes in the network is N_{new_node}, then the time needed to bind all of them (T_{all_bind}) is

$$T_{all\_bind} = N_{new\_node} \times T_{a\_bind} \quad (9)$$

The data field of switch mode global frame is 2 bytes, that of the *switch mode selective* is 5 bytes, the response frame is 2 bytes, configure Node-ID is 2 bytes and that of the response frame is 3 bytes. The identifier of all frames is 29 bits. By considering the worst case transmission times on the CAN bus, the two bytes data field frames are 165 bits, three bytes data field frames are 175 bits, five bytes data field frames are 195 bits. If the number of new discovered nodes (N_{new_node} is 30, the time taken to bind the nodes (T_{all_bind}) is 1.15s.

In order to prove the correctness and the performance of our technique, a network analyzer for CAN systems which also supports virtual node emulations is utilized to verify the binding protocol. The CANoe tool is used to trace the frame flows over the bus. Real time messages generated by CAPL block of the virtual nodes is recorded in log files, which reflects the current states and emulation results of the virtual system. Thus, the correctness and the performance of the binding protocol could be both verified. In Fig. 9, a screenshot of the output obtained by CANoe is present. The trace window contains 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.

We emulate a CAN network which contains a virtual master node and 120 virtual slave nodes, part of nodes are show as Fig. 8. The behavior of master and slave nodes is described in the Communication Access Programming Language (CAPL), a C-like programming language. Both the master and slave nodes are RAND-compliant, and each virtual slave is assigned with a 16 bytes LSS address. The result of our emulation experiment shows that all slave nodes can be discovery efficiently and be signed a proper Node-ID.

In order to demonstrate our method more explicitly, we consider a network with one master and five slave nodes.

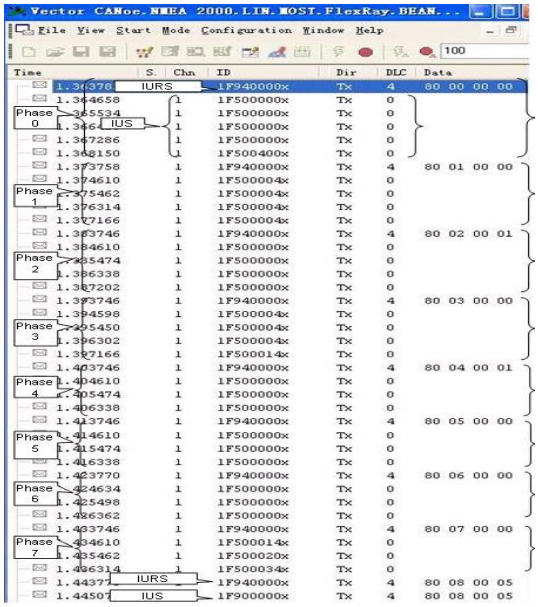

Figure 8. Part of nodes in emulated can network

In Node-IDs pre-assigning phase, Node_Table which is maintained by the master is filled as table 1. This indicates that there are three types of slave nodes in the network, and type codes of the slave nodes are 0x**01000001**00000001 (A), 0x**00000001**00000001 (B) and 0x**00000001**00000005 (C) respectively. The number of slave nodes with type code as B is three, A and C are one. Slave nodes with type code A must be connected to the network through slot 1, C through slot 3, and B through slot (2,4,5). In discovery and binding phase, after discovery step is completed, the master node collects LSS addresses of all the five slave nodes, and the LSS addresses are temporarily store in array Discovered_New_Node ascendingly. In the binding step, the collected LSS addresses are filled in Node_Table according to the given algorithm (Table 2). We use *s1* to represent the slave node with LSS address 0x**01000001**00000001000000000000001, *s2*, *s3*, *s4*, *s5* are similar. According to the given rules, the slave node s1 should have been connected to the network through slot 1, and s2, s3, s4, s5 through slot 2, 3, 4, 5 respectively.

The detail process of frames exchanging between the master and slave nodes is shown as Fig. 9. The master issues a *IURS* (phase 0) to start a node discovery process, and all slave nodes response a *IUS* frame. In the phase 1, the master sends a *IURS* whose data field is (80 01 00 00), and this indicates that the slave nodes whose chunk(0) of the LSS address is (00 00) will be selected. The chunk(0) of the slave node s1 is (01 00), so *s1* switches its status to *unknown*. The chunk(0) of the others nodes match (00 00), they switch their status to *selected* and issue the responding IUS frame. For chunk(0),chunk(1) and chunk (2) of slave nodes *s2*,*s3*,*s4*,*s5* are equivalent, they all issue *IUS* frame to response the *IURS* frame in phase 1, 2 and 3. In phase 4, the data field of *IURS* is (08 04 00 01). The slave node s3 switches its status to *unknown* and the slave nodes *s2*, *s4*, *s5* continue to keep their status *selected*. In phase 5, 6 and 7, the slave nodes *s2*, *s4* and *s5* can't be made a distinction for their corresponding chunks are equal. After phase 7, the master issues the *DIRS* frame whose data field is (80 08 00 05), and only the slave node *s2* is selected. The slave node *s2* is detected by the master and switches its status to *identified*. The status of *s4* and *s5* will be switched to *unknown*. The slave nodes *s1*, *s3*, *s4* and *s5* will participate in the next slave node discovery process for their statuses are all *unknown*.

After the discovery step is ended and the Node_Table is filled, the LSS protocol is utilized to configure Node_ID for slave nodes. The detail process of frames exchanging between the master and slave nodes is shown as Fig. 10. The master issues a *switch mode global frame* to switch all slave nodes to operational mode first. Second, four *switch mode selective* frame are issued by the master to switch the slave node s1 to *configure* mode, and s1 issues a response frame after be selected by the master. Last, the master issues the *configure Node ID* frame to assign Node-ID 1 to the slave node *s1*, and *s1* confirms the assignment. Similarly, the slave node *s2*, *s3*, *s4* and *s5* can be configured.

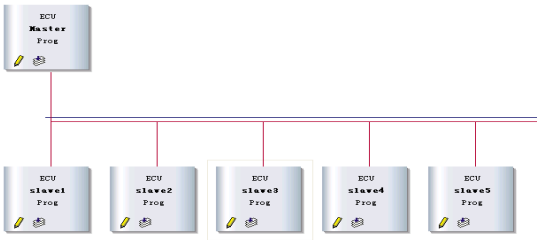

Figure 9. A LSS address discovery process of RAND

We disable slave nodes *s2* and *s4* to emulate the scene that *s2* and *s4* are failed. In discovery and binding phase, after the preparation step, the R_SN and isOnline fields of rows 2 and 4 are cleared. We change the LSS addresses of slave nodes *s2* and *s4*. Now, the R_SN of s2 is 0x0000000000000015, and s4 is 0x0000000000000018. After the LSS addresses are modified, the slave nodes s2 and s4 are enabled again, and this process emulate that slave nodes are replaced by well working devices. In discovery and binding phase, after discovery step is completed, the two new LSS addresses are collected. In the binding step, the new collected LSS addresses are filled into Node_Table and the slave nodes s2 and s4 are configured Node-ID again.

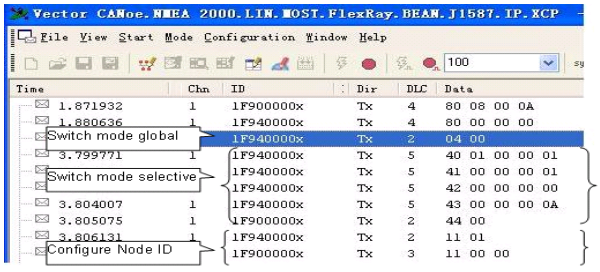

Figure 10. A Node-ID configure process

V CONCLUSION

CANopen is an application protocol conceived for distributed control systems that adopts CAN system to carry out frame exchanges. Currently, Node-ID assignment can be accomplished by means of LSS services, which only provide a mechanism to configure the Node-ID for a slave node, but are not competent for configuring Node-ID for all slaves automatically. Additionally, the LSS address of the slave node must be known to the master before the Node_IDs been configured. We applied the LSS physical address structure to the AND protocol and made some necessary corresponding modifications. The revision of the AND protocol is proved to be able to detect LSS address of all slaves efficiently. In our proposed protocol, the Node_ID assignment is divided into two phases. First, node types are mapped to Node-ID clusters. This is done in the master node before slave nodes are physically connected to the network. Then, LSS addresses of all slave nodes are collected by the master node, and Node-IDs are assigned to slaves with the same rule as slave nodes connected to the network. In fact, a Node-ID can also reflect the logic position of a physical slave node. Finally, the performance of the protocol is evaluated both from a theoretical point of view and by means of a network emulator. The latter approach also verified the correctness of the protocol under actual conditions.

ACKNOWLEDGMENT

This study was supported by National Natural Science Foundation of China (No. 60674081 and No.61074145), Independent Innovation Research Foundation of Huazhong University of Science and Technology, China(No. 2011TS029).

REFERENCES

[1] ISO 11898, "Road Vehicles-Interchange of Digital Information -Controller Area Network (CAN) for High-Speed Communication," 1993.
[2] G. Zheng-Nan, C. Qing-Wei, H. Wei-Li, "A new experimental platform for networked control systems based on CAN and switched-Ethernet," Information Technology Journal, vol. 10, no. 1, 2011, pp. 219-230.
[3] X. Yu, H. Gao, Z.Deng, "The research on control of lunar rover with rocker bogie based on bus network driving," Information Technology Journal, vol. 7, no. 7, 2008, pp. 1001-1008.

TABLE II.
THE CONTENTS OF NODE_TABLE AFTER THE BINDING STEP

Node_ID	Type_Code	R_SN	isOnline
1	0x**01000001**00000001	0x000000000000000A *(s1)*	1
2	0x**00000001**00000001	0x0000000000000005 *(s2)* 0x0000000000000015 (replace)	1
3	0x**00000001**00000005	0x0000000000000008 *(s3)*	1
4	0x**00000001**00000001	0x0000000000000008 (s4) 0x0000000000000018 (replace)	1
5	0x**00000001**00000001	0x000000000000000D (s5)	1

[4] CiA, "DS301, Version 4.02, CANopen Application Layer and Communication Profile," 2002.
[5] CANopen, 50325-4, "Industrial Communications Subsystem Based on ISO 11898 (CAN) for Controller-Device Interfaces Part 4," 2002.
[6] CiA, "DSP 305, Version 1.1," Layer Setting Services and Protocol (LSS), 2002.
[7] K. W. Tindell, H. Hansson, A. J. Wellings, "Analysing real-time communications: Controller Area Network (CAN)," Proc. Real-Time Systems Symp., 1994, pp. 259-263.
[8] K. W. Tindell, A. Burns, A. J. Wellings, "Calculating Controller Area Network (CAN) messages response times," Control Eng. Practice, vol. 3, no. 8, 1995, pp. 1163-1169.
[9] CiA, "DS204-1, CAN Application Layer for Industrial Applications: DBT Service Specification," 1996.
[10] CiA, "DS204-2, CAN Application Layer for Industrial Applications: DBT Protocol Specification," 1996.
[11] S. Cavalieri, "A protocol for dynamic assignment of identifiers in CAN application layer," in Proc. 4th IFAC Int. Conf. Fieldbus Systems and Their Applications, Nancy, France, 2001, pp. 119-126.
[12] R. Droms, "Dynamic Host Configuration Protocol," RFC1541," Bucknell University, 1993.
[13] G. Cena, A. Valenzano, "A protocol for automatic node discovery in CANopen networks," IEEE Trans. on Industrial Electronics, vol. 50, no. 3, 2003, pp. 419-430.

Guangcan Yu was born in Hubei, China, in 1976. He is a postdoctor in the Department of Control Science & Engineering at the Huazhong University of Science and Technology. He received the Ph.D. degrees in the College of Computer Science & Technology from Huazhong University of Science and Technology, Wuhan, China, in 2008.

He research interests include network protocol design, artificial intelligent and industrial communication. He current research focuses on safty and security for industrial control system.. Contact he at Email:ygcan@QQ.com.

Chunjie Zhou was born in Hubei, China, in 1965. He received the M.S. and Ph.D. degrees in control theory and control engineering from. Huazhong University of Science and Technology, Wuhan, China, in 1991 and 2001, respectively. He is currently a doctoral tutor professor in the department of control science and engineering at Huazhong University of Science and Technology. His research interests include fault detection and diagnosis, fault-tolerant control, artificial intelligent, industrial communication.

Shuang Huang was born in Hubei, China, in 1986. He received the BS degree in Automation from Huazhong University of Science & Technology of Control Science and Engineering, Wuhan, China, in 2009, He is currently working toward the PhD degree in Control Science and Engineering from Huazhong University of Science and Technology. His research interests include industrial communication and theory& security for industrial control system.