

## **Copyright Warning & Restrictions**

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

**Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation**

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

## **ABSTRACT**

### **PETRI NET MODELING AND PERFORMANCE ANALYSIS OF CAN FIELDBUS**

**by  
Ji-Qiong Zhou**

The CAN FB (Controller Area Network FieldBus) has been in existence for ten years. It supports automated manufacturing and process control environments to interconnect intelligent devices such as valves, sensors, and actuators. CAN FieldBus has a high bit rate and the ability to detect errors. It is immune to noise and resistant to shock, vibration, and heat. Two recently introduced mechanisms, Distributed Priority Queue (DPQ) and Priority Promotion (PP) enable CAN FieldBus networks to share out the system bandwidth and grant an upper bound on the transmission times so as to meet the requirements in real-time communications. Modeling and analysis of such networks are an important research area for their wide applications in manufacturing automation.

This thesis presents a Petri net methodology which models and analyzes CAN FieldBus access protocol. A Reachability Graph of the Petri net model is utilized to study the behavioral properties of the protocol. A timed Petri net simulator is used to evaluate the performance of the protocol. Performance measures include the completion time for successful events and operations. Operational parameters investigated using the Petri Net model are FieldBus speed, the length of each frame, and the number of frames in a message.

**PETRI NET MODELING AND PERFORMANCE  
ANALYSIS OF CAN FIELDBUS**

**by  
Ji-Qiong Zhou**

**A Thesis  
Submitted to the Faculty of  
New Jersey Institute of Technology  
in Partial Fulfillment of the Requirements of the Degree of  
Master of Science in Electrical Engineering**

**Department of Electrical and Computer Engineering**

**May 1998**

Blank Page

**APPROVAL PAGE**

**PERTRI NET MODELING AND PERFORMANCE  
ANALYSIS OF CAN FIELDBUS**

**Ji-Qiong Zhou**

---

Dr. Mengchu Zhou, Thesis Advisor Date  
Associate Professor of Electrical and Computer Engineering, NJIT

---

Dr. John Carpinelli, Committee Member Date  
Associate Professor of Electrical and Computer Engineering, NJIT

---

Mr. Phillip Ong, Committee Member Date  
Senior Technical Staff Member, AT&T Laboratories, Holmdel, NJ

## **BIOGRAPHICAL SKETCH**

**Author:** Ji-Qiong Zhou

**Degree:** Master of Science

### **Undergraduate and Graduate Education:**

- Master of Science in Electrical Engineering, 1998  
New Jersey Institute of Technology, Newark, New Jersey
- Bachelor of Engineering in Telecommunications Engineering,  
Nanjing University of Posts & Telecommunications, 1982

**Major:** Electrical Engineering

To my family



## ACKNOWLEDGMENT

I would like to express my greatest appreciation to Dr. Mengchu Zhou for his advice and guidance throughout the progress of this thesis.

Special thanks to Dr. John Carpinelli and Mr. Phillip Ong for their important contribution as members of the committee.

The deepest gratitude is extended to my family, particularly to my father, who has helped me to become a mature engineer and encouraged me to pursue further education for an advanced degree.

The guidance and support of Dr. Michael Zhou, my brother, was essential to achieve this goal, to him I owe my eternal gratitude.

Finally, but not the least, I would also like to thank Dr. Venky Kurapati for his valuable help.

## TABLE OF CONTENTS

<b>Chapter</b>	<b>Page</b>
1 INTRODUCTION .....	1
1.1 Origin of FieldBus.....	1
1.1.1 FieldBus Defined.....	1
1.1.2 Characteristics of FieldBus Networks.....	2
1.1.3 Advantages of FieldBus Networks.....	3
1.1.4 New Functionalities of FieldBus Networks .....	3
1.1.5 Field Devices .....	4
1.2 Evolution of CAN FieldBus .....	5
1.3 Objective. ....	5
1.4 Organization .....	6
2 LITERATURE REVIEW .....	7
2.1 Research and Applications in the FieldBus.....	7
2.2 CAN FieldBus .....	8
2.2.1 Origin of CAN FieldBus .....	8
2.2.2 Evolution of CAN FieldBus.....	9
2.2.3 DPQ and PP of CAN FieldBus.....	11
2.3 Petri Net Theory and Applications.....	14
2.3.1 Definition.....	14
2.3.2 Timed PNs.....	23

**TABLE OF CONTENTS**  
**(Continued)**

<b>Chapter</b>	<b>Page</b>
3 CAN FIELDBUS .....	24
3.1 FieldBus.....	24
3.1.1 FieldBus Model and Specifications.....	24
3.1.2 Two Speed FieldBuses .....	27
3.2 CAN Model .....	29
4 MODELING AND ANALYSIS BY USING PNS.....	33
4.1 Modeing CAN FieldBus with Petri Nets.....	33
4.1.1 Modeing Methodology .....	33
4.1.2 The Modeing of Four Primitives and Concepts with Petri Nets .....	35
4.1.3 Modeing Example for the System.....	36
4.2 Performance Analysis of CAN FieldBus .....	41
4.2.1 Petri Net Properties .....	41
4.2.2 Reachability and Petri Net.....	42
4.3 The System Process Time Computation .....	43
5 SIMULATION.....	57
5.1 A Timed Petri Net Simulator.....	57
5.2 Simulation of the Petri Net Model .....	59
5.3 Simulation Results.....	59

**TABLE OF CONTENTS**  
**(Continued)**

<b>Chapter</b>	<b>Page</b>
6 CONCLUSION.....	65
6.1 Summary .....	65
6.2 Future Work .....	66
REFERENCES .....	69

## LIST OF TABLES

<b>Table</b>	<b>Page</b>
2.1 Graphical Representation of Petri Net Components.....	17
4.1 Source and Destination Place Description.....	53
4.2 Source and Destination Transition Description and Time Delay .....	54
4.3 The Relationship among FieldBus Speed the Length of the Frame and Time Delay in $t_2$ .....	55
4.4 Loops and Their System Processes, Time Required to Complete the Process and Cycle Time (Speed=2.5Mb/s, 64bits/frame).....	56
5.1 Effect of the FieldBus Speeds (or $t_2$ Delays) on Cycle Time ( $k=4$ , and 64bits/frame) .....	62
5.2 Effect of the Values of the Counter and the FieldBus Speeds (or $t_2$ Delay) on Cycle Time (128bits/frame).....	63
5.3 Effect of the Lengths of the Frame (or $t_2$ ) on Cycle Time .....	64

## LIST OF FIGURES

<b>Figure</b>	<b>Page</b>
2.1 Frame Format for CAN .....	10
2.2 Frame Format for DPQ & PP.....	12
2.3 DPQ/PP Sequence of Operations.....	13
2.4 Petri Net Execution Rules .....	18,19
2.5 Example of Marking and Firing Transition .....	21
3.1 OSI and FieldBus Models.....	30
3.2 Frame Format at the Data Link Layer of the FieldBus Network.....	31
3.3 Lower and Higher Speed FieldBuses.....	32
4.1 The Petri Net Model for the CAN FieldBus Access Protocol(Counter=k) .....	45
4.2 Representation of Four Primitives .....	46
4.3 Reachability Graph for the CAN FieldBus Access Protocol (k=1) (with only marked places shown in each marking) .....	47
4.4 Reachability Graph for the CAN FieldBus Access Protocol (k=1) .....	48
4.5a) Transition Sequence for the CAN FieldBus Access Protocol, Part 1 (k=2) .....	49
4.5b) Transition Sequence for the CAN FieldBus Access Protocol, Part 2 (k=2).....	50
4.6 Reachability Graph for the CAN FieldBus Access Protocol (k=2) .....	51
4.7 Effects of Some Factors on a Successful Communication (k=1, Speed=2.5Mb/s and 64bits/frame) .....	52
5.1 The Petri Net of the System for Simulation.....	60
5.2 Effect of $t_2$ Delays on Cycle Time (k=4, 64bits/frame) .....	62

**LIST OF FIGURES**  
**(Continued)**

<b>Figure</b>	<b>Page</b>
5.3 Effect of the Values of the Counter and $t_2$ Delays on Cycle Time (lgx) (128bits/frame) .....	63
5.4 Effect of $t_2$ Delays on Cycle Time (k=4, Speed=2.5Mb/s) .....	64

# CHAPTER 1

## INTRODUCTION

### 1.1 Origin of FieldBus

Industry still lacks a common digital interface for connecting field devices (sensors, valves, pumps, temperature transmitters, and so on) to a system controller. The 4-20mA analog current loop is still the only widely accepted standard for implementing process control systems. It constrains the information a controller can receive from an instrument to a single reading that reflects the instrument's primary process value. To overcome such limitations, a FieldBus is now proposed as an industry wide standard for a variety of business reasons. Its key functional aspects ensure that field devices can easily share data and execute standard functions in a control system. Currently, more than 40 FieldBus standards are commercially available all over the world [3].

The first ideas for a FieldBus emerged from the nuclear, military, and instrumentation applications giving birth to the following communications systems: CAMAC, MIL-STD 1553, and IEEE 488; however, only the MIL-STD 1553 is based on serial communications thus capable of meeting the requirements of a FieldBus. More recently, several proposals for standardization have appeared, such as FIP, Bitbus, Proway, Profibus [4], and CAN [5].

#### 1.1.1 FieldBus Defined

The FieldBus is an all-digital, serial, and two-way communication network which interconnects field devices (sensors, actuators, pumps, valves, temperature transmitters,



and so on) to a system controller. FieldBus is a Local Area Network (LAN) for instruments used in both manufacturing automation and process control with built-in capability to distribute the control application across the network. Since FieldBus supports communications at the lowest level of the automated manufacturing control hierarchy, i.e., the interconnection of intelligent devices with their sensors and actuators, FieldBus networks are also called *sensor level networks*.

FieldBus is an enabling technology allowing suppliers to develop products which can operate with one another as well as with control systems. Because FieldBus moves data much faster than today's hybrid protocols do, it is easily networked over a shared bus. There are two kinds of FieldBus according to its speed: a lower-speed FieldBus (H1), and a higher-speed one (H2). H1 operates at 31.25kb/s, and H2 runs at 1 or 2.5 Mb/s.

FieldBuses can be considered as a special class of local area networks (LAN's) designed to satisfy the needs of real-time communications in the areas of industrial automation and process control [5]. A well-designed FieldBus should be able to ensure both a bounded access time to each device for real-time (RT) traffic and a fair shareout of system bandwidth among the different nodes for non-RT traffic.

### **1.1.2 Characteristics of FieldBus Networks**

1. Short messages: Ability to process very short messages in an efficient manner.
2. Periodic and aperiodic traffic: Ability to process periodic and aperiodic traffic.

Periodic traffic is due to sampled data encountered in most computer based control

systems. Aperiodic traffic is due to asynchronous events and condition changes such as a conveyor failure and arrival of a raw material piece.

3. Short response times: Bounded and sometimes fixed response times. This is a consequence of the periodic traffic, and real time requirements of some event conditions such as alarm messages.
4. Good reliability and safety.
5. Low network interface cost: Communications is serial so as to save cable cost.

### **1.1.3 Advantages of FieldBus Networks**

FieldBus networks have the following advantages [4]:

- a) All costs can be significantly reduced by replacing many wires using a single wire as the network medium (i.e., serial communications), and by using simpler network architectures and protocols than networks at other levels of the control hierarchy.
- b) Easier installation and maintenance, because the operations involve the manipulation of a unique cable.
- c) Easier detection and localization of cable faults.
- d) Easier expansion because of the modular nature of the network.

### **1.1.4 New Functions of FieldBus Networks**

FieldBus networks bring new functionalities [4]:

- a) Data consistency: All network stations are interconnected by a broadcast channel in a reliable fashion.

- b) Better noise immunity: Because of the digital communication, the signal to noise ratio can be improved by using more bits to represent an analog value.
- c) Preprocessing capability: Before the samples are sent over the network, each station has some processing capabilities that could perform some preprocessing operations.

### **1.1.5 Field Devices**

Today there are three categories of field devices: traditional analog and discrete I/O; hybrid analog and digital devices; and purely digital devices.

1. Devices with traditional analog and discrete I/O connect to an I/O system or controller by means of dedicated 4—20-mA analog current loops. They are strictly point-to-point systems where each device has its own connection to a host controller. Examples include analog temperature and pressure transmitters.
2. Hybrid analog and digital devices are used with both analog and digital communications systems. For instance, the well-established HART (highway addressable remote transducer) superimposes a digital communication signal on the top of a standard 4—20-mA analog signal. The digital signal is processed to have a zero average value so as not to affect suitably averaged readings of the analog current.
3. Purely digital devices often have open interfaces. However, they usually require custom interfacing hardware and need the use of customized software drivers in a control system. The devices include weigh scales, bar-code readers, process analyzers, supervisory control and data acquisition systems, programmable logic controllers, and some intelligent valves with digital outputs.

## **1.2 Evolution of CAN FieldBus**

CAN FieldBus is network protocol which is initially developed to support cheap and rather simple automotive applications. Because of its high performance, simplicity, and low cost, it has recently been used to interconnect intelligent devices for automated manufacturing and process control environments. After Distributed Priority Queue (DPQ) and Priority Promotion (PP) mechanisms was introduced in 1997 [5], CAN has the ability to ensure either a deterministic bound on the transmission times experienced by the RT communications or a fair shareout of the network bandwidth for the non-RT connections. These improvements have been obtained through a dynamic reassignment of identifier field in the exchanged frames. DPQ and PP mechanisms use different ways to reassign priorities dynamically, respectively. When the traffic increases and approaches the network bandwidth or, even worse, when it exceeds the capacity of the bus, both DPQ and PP will be able to fairly control the access to the communication channel. DPQ and PP are very simple and can be implemented at very low cost.

## **1.3 Objective**

This thesis evaluates the CAN FieldBus access protocol by using a Petri net modeling and analysis method. CAN FieldBus is a carrier sense multiple access (CSMA) network. This protocol has wide applications in industrial automation.

It is recognized that there are three parameters which influence significantly the performance of the CAN FieldBus protocol:

- 1) Speed in sending and receiving messages;
- 2) Number of frames in a message;

3) Length of each frame.

Petri nets (PNs), especially deterministic timed Petri nets, are suited to the modeling of the system and evaluation of the services and operations of the CAN FieldBus protocol. Petri net model is developed for the entire system. A reachability graph of the Petri net model is used to research the behavioral properties of the system. Assuming deterministic delays, we perform temporal analysis and calculate the completion time for successful events and operations under different parameters.

#### **1.4 Organization**

This thesis is organized as follows. The next chapter simply describes two modifications of the basic CAN FieldBus (CAN FieldBus priority control mechanisms of DPQ and PP) and provides basic concepts, properties and applications of Petri nets. Chapter 3 describes types of protocols in CAN FieldBus and CAN FieldBus applications. Chapter 4 presents the Petri net model of CAN FieldBus access protocol and the performance analysis based on the TimedPNT model. Chapter 5 describes the deterministic TimedPNT software package that is used as a simulation tool, and presents the simulation results. Finally, Chapter 6 summarizes the contributions of this thesis and discusses the future work.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Research and Applications in the FieldBus

- *Application of the FieldBus Protocol:*

An evaluation of the application of the Normalized Proportional Allocation Scheme (NPAS) to the asynchronous bandwidth allocation of the FieldBus protocol is presented [9]. The analysis is focused on the gain that can be achieved by using the NPAS for bandwidth allocation, after deriving the suitable asynchronous bandwidth size that will keep the token cycle time within a certain bound. The gain is compared with the case when the Equal Partition Allocation Scheme (EPAS) is used, considering an asymmetrically loaded system. The simulation results show the superiority of using the NPAS over the EPAS, in terms of increasing channel utilization, and decreasing transfer delays for less non-urgent traffic [9].

- *Enhancing Performance in FieldBus Communication Systems:*

Since there is the problem of communication in centralized medium access control FieldBus systems, a management policy is presented for periodic traffic [14]. It allows the respect of the asynchronous time-constraints to be enhanced. The policy proposed is modeled and simulated. The results show the improvement in performance that can be obtained by using the proposed policy.

- *Centralized versus Distributed Protocols for FieldBus Applications:*

FieldBus communication systems provide a more efficient interconnection of field devices in Factory Automation environments than was possible with traditional point-

to-point systems. A critical point in FieldBus communication systems, however, is represented by access to the distributed physical medium; and the use of one Data Link Layer (DLL) access protocol rather than another may be decisive in order to reach the optimal performance levels. The solutions adopted at this level may be subdivided into two categories: centralized (used in IEC/ISA FieldBus and FIP protocols) and distributed control protocol (used in Profibus protocol). Analyzing the centralized and distributed approaches used in the IEC/ISA and Profibus protocols is to assess the type of compatibility between them. Then, their performance is evaluated through simulation based on Petri net modes of the protocols [13].

- *Calculating CAN Message Response Times:*

Controller Area Network (CAN) is a well-designed communications bus for sending and receiving short real-time control messages. One of the perceived drawbacks to CAN has been the inability to bound accurately the worst-case response time of a given message (i.e., the longest time between queuing the message and the message arriving at the destination processors). An analysis is presented to bound such latencies [12].

## **2.2 CAN FieldBus**

### **2.2.1 Origin of CAN FieldBus**

CAN was initially developed for the automotive industry in order to solve the cabling problems found in some kinds of vehicles [5]. Since it is highly versatile in industrial application, it can also be adopted as a control network. CAN operates by enabling any device on the bus to broadcast a message packet over the bus. Since the basic CAN can

neither ensure deterministically bounded transmission times for RT data exchanges nor a fair shareout of the network bandwidth among non-RT application processes, the basic CAN is not able to satisfy all the communication needs in the industrial automation and process control.

### **2.2.2 Evolution of CAN FieldBus**

Two modifications of the basic CAN access protocol are done by introducing to them distributed priority queue (DPQ) and priority promotion (PP). They are different priority control mechanisms of CAN, which enable CAN network to meet the above two requirements, i.e., bounded transmission times for RT traffic and fair shareout of the bandwidth for non-RT traffic.

CAN is derived from the CSMA/CD channel access technique. Just like the CSMA/CD networks, CAN adopts a common bus topology [5]. Most of the features of CAN concerning the bandwidth shareout and access delays mainly depend on two mechanisms adopted on the data-link layer (DLL) which is split up into two sublayers, namely, MAC (medium access control) and LLC (logical link control). The priorities in CAN is important in resolving conflicts if more than one station start transmitting at the same time. All stations on the network receive the message, based on its priority rating, and can determine whether to act on the information. If a station initiates a message successfully (or a possible contention is solved), the medium access mechanism of CAN is allowed to continue until all the message is sent out. LLC sublayer is used for deciding whether or not the received frames are relevant to the station. The LLC sublayer also



manages the retransmission of those frames which are lost in a contention or which have been involved in transmission errors.

***Frame Format for CAN:***

The CAN network uses an identifier field (IF) in each frame to support real-time communications. Both the LLC and MAC sublayers use this information in IF field with different meanings and purposes (Fig. 2.1). Fig. 2.1 shows the format of a basic CAN frame. The start of frame (SOF) field consists of a single dominant bit, while the identifier field and the remote transmission request (RTR) bit form the arbitration field, which is used by the MAC mechanism to resolve any contention on the bus. The CRC field is a cyclic redundancy check on the frame content. The ACK field is overwritten by receiving nodes to acknowledge the correct reception frame.

The MAC sublayer uses this IF content as the priority level of the current frame and resolves in a nondestructive way any collision of messages on the bus. On the other hand, LLC sublayer uses the IF information to recognize the transmitted objects and dispatch them to the application programs.

S O F	11 bit Identifier	R T R	Control Field	Data Field	CRC Field	ACK Field	End of Frame
-------------	----------------------	-------------	------------------	---------------	--------------	--------------	-----------------

**Figure 2.1** Frame Format for CAN

### 2.2.3 DPQ and PP of CAN FieldBus

Since each connection has a different (and fixed) priority and the object identifiers are assigned in a static way, services for transferring a large amount of data can block other applications for long time periods. If a minimum time is defined between any two subsequent transmissions, the wasted network bandwidth is increased. To adapt the ordering of transmissions to the timing requirements of the applications, a dynamic reassignment of the value in IF is a fair practice in a CAN network.

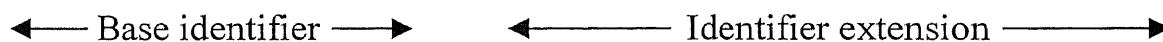
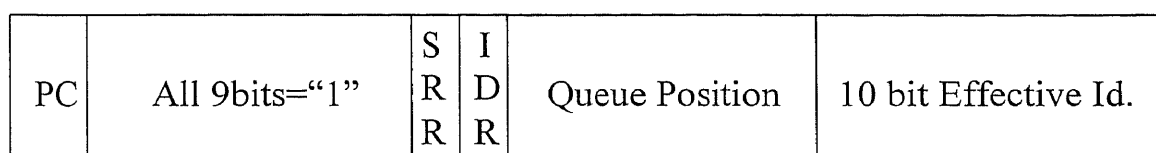
Two mechanisms (DPQ and PP) are used to reassign priorities dynamically so as to improve the fairness of a CAN network significantly, and offer maximum flexibility to the applications in selecting the most suitable priority for their data communications (Figure 2.2). The priority class (PC) field shown in Fig. 2.2 encodes the service priority. Four classes are defined: time critical (highest), high, low, and time available (lowest). The SRR stands for substitute remote request, and it is a padding field for extended frames. Its value must be 1 for extended frames. IDE bit means when equal to 0 denotes a standard 11 bit identifier frame, when equal to 1 denotes 29 bit extended frame.

**DPQ:** The DPQ mechanism is based on a global distributed FIFO queue, where all nodes using this arbitration scheme are virtually inserted [5].

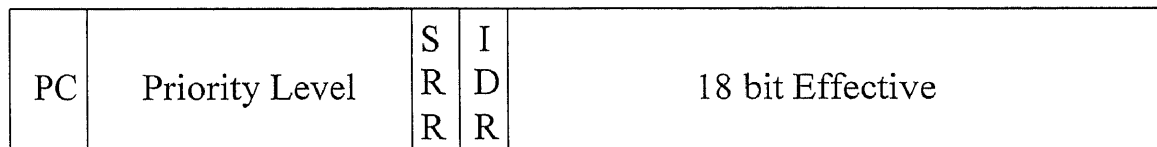
**PP:** The priority promotion arbitration scheme is based on the idea of assigning each station a priority level which is progressively increased proportionally to the time elapsed since the station first attempts to transmit the current frame [5]. Each time a station experiences a collision with other nodes transmitting frames at the same priority class and loses the contention, it raises its priority level. By contrast, after a successful transmission, the sending station sets it to the lowest priority. PP does not require

different stations to be assigned different priorities. When two or more nodes with the same priority level begin their transmissions at the same time, the contention is resolved by considering the effective identifier field.

Both DPQ and PP base their operations directly on the standard MAC services, use the information conveyed in the MAC indication, and confirm primitives to update the local station priority as shown in Figure 2.3.



*DPQ*



*PP*

**Figure 2.2** Frame Format for DPQ & PP

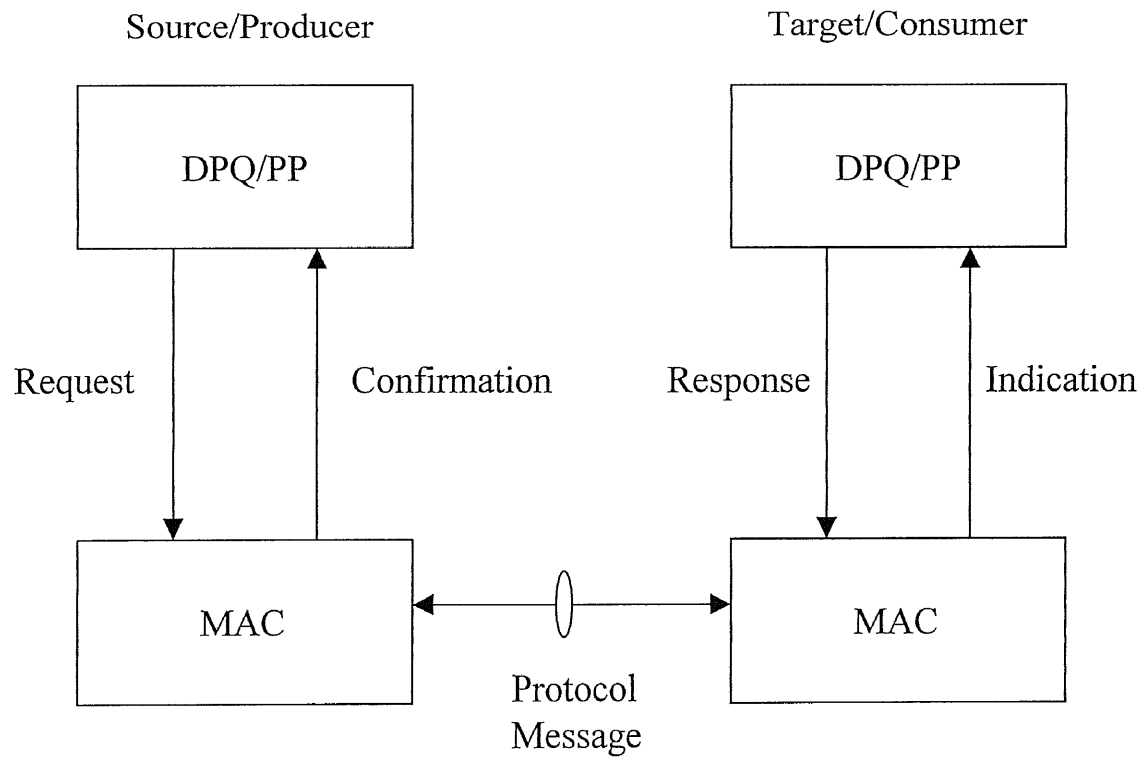


Figure 2.3 DPQ/PP Sequence of Operations

### 2.3 Petri Net Theory and Applications

Petri nets (PNs) were named after Carl A. Petri who created in 1962 a net-like mathematical tool for the study of communication with automata. Their further development was facilitated by the fact that Petri nets can be used to model properties such as process synchronization, asynchronous events, concurrent operations, and conflicts or resource sharing [6]. Petri nets are a graphical and mathematical modeling tool to many systems. PNs are fruitfully applied in various important fields such as computer systems and networks, and manufacturing systems. Useful classes of PNs include Ordinary (Untimed) PNs, Timed PNs, and Stochastic PNs.

#### 2.3.1 Definition

A marked Petri net  $Z=(P, T, I, O, M)$  is a five tuple; where

1.  $P=\{p_1, p_2, \dots, p_n\}$ ,  $n>0$ , is a finite set of places;
2.  $T=\{t_1, t_2, \dots, t_n\}$ ,  $s>0$ , is a finite set of transitions,  $P \cup T \neq \emptyset$ , and  $P \cap T = \emptyset$ ;
3.  $I: P \times T \rightarrow \mathbb{N}$ , is an input function that defines directed arcs from  $P$  to  $T$ , where  $\mathbb{N}=\{0,1,2,\dots\}$ ;
4.  $O: P \times T \rightarrow \mathbb{N}$ , is an output function which defines directed arcs from  $T$  to  $P$ ;
5.  $M: P \rightarrow \mathbb{N}$ , is a marking whose  $i$ th component represents the number of tokens in the  $i$ th place. An initial marking is denoted by  $M_0$ . Tokens are pictured by dots.

The four tuple  $(P, T, I, O)$  is called a PN structure that defines a directed graph structure.

***Enabling Rule:***

A transition is enabled if each input place  $p$  of  $t$  contains at least the number of tokens equal to the weight of the directed arc connecting  $p$  to  $t$ . This means that if all the input places of transition have enough tokens, then  $t$  is enabled. Mathematically, transition  $t \in T$  is enabled if and only if  $m(p) \geq I(p, t), \forall p \in P$ .

***Firing Rule:***

Enabled in marking  $M$ ,  $t$  fires and results in a new marking  $M'$  following the rule:  $M'(p) = M(p) - I(p, t) + O(p, t), \forall p \in P$ . Marking  $M'$  is said to be immediately reachable from  $M$ . This says that when all the input places hold enough number of tokens, an event embedded in a transition can happen, an enabled transition  $t$  fires and its firing removes  $I(p, t)$  tokens from  $p$ , and then deposits  $O(p, t)$  tokens back into  $p$ .

***Components of PNs:***

A Petri Net comprises a set of Places ( $P$ ) and Transitions ( $T$ ) which are linked to each other by Arcs.

- *Places:* The places are used to represent condition and/or status of a component in a system. They are pictured by circles.
- *Transitions:* The transitions represent the events and/or operations. Empty rectangles or solid bars represent transitions.

A transition can be immediate or timed in terms of their delays and sequential, concurrent, or in conflict with respect to others.

- *Immediate transitions:* Immediate transitions fire as soon as they are enabled. Immediate transitions are represented by bars.

- *Timed transition:* In a timed transition, there is a delay between enabling and firing. Timed transitions are represented by rectangular boxes and can be either random (stochastic) or deterministic.

In case of a conflict, since they share the same input place, the enabled transitions may not fire simultaneously. When two or more transitions can fire simultaneously, they are called concurrent transitions.

- *Arcs:* An arc represents information control or material flow.
- *Inhibitor Arc:* A transition associated with the inhibitor arc cannot fire if the place to which it links contains a token. Graphically, an inhibitor arc ends with a small circle at a transition. Generally, we use an inhibitor arc as a safety feature to prevent execution of a particular task or action.
- *Self-loop Arc:* A self-loop arc puts the token back to the same place after the transition is fired. They can be eliminated from the analysis view point.
- *Tokens:* Tokens reside in places and are used to specify the PN state (also called the marking). Table 2.1 illustrates the graphical representation of the above mentioned components.





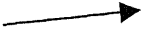
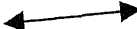
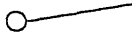

***Examples:***

The enabling and firing rules are illustrated in Fig. 2.4. In Fig. 2.4(a), transition  $t_1$  is enabled as the input place  $p_1$  of transition  $t_1$  contains at least two tokens and the input place  $p_2$  of transition  $t_1$  contains at least three tokens. Since  $I(p_1, t_1)=2$  and  $I(p_2, t_1)=3$ . The firing of the enabled transition  $t_1$  removes from the input place  $p_1$  two tokens as  $I(p_1, t_1)=2$  and removes from the input place  $p_2$  three tokens as  $I(p_2, t_1)=3$ , and then deposits two tokens in the output place  $p_3$  since  $O(p_3, t_1)=2$ , and one tokens in the output place  $p_4$

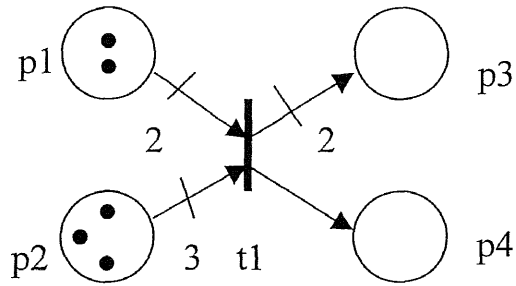
since  $O(p_4, t_1)=1$ . This is pictured in Figure 2.4(b). In Fig. 2.4(c) and (d), the enabling and firing rules are similar to the former examples. An important difference is that the self-loop arc puts two tokens back to the place  $p_1$  after the transition  $t_1$  is fired.

We use the presence of the inhibitor arcs connecting the input place  $p_2$  to the transition  $t_1$  to change the transition  $t_1$ 's enabling conditions. For example, in Fig. 2.4(e), since the place  $p_2$  contains the number of tokens less than the weight of the arc, the transition  $t_1$  is enabled. The firing, however, does not change the marking in the inhibitor arc connected place  $p_2$ . In Fig. 2.4(f), the place  $p_2$  contains three tokens equal to the weight of the arc, so the transition  $t_1$  is not enabled.

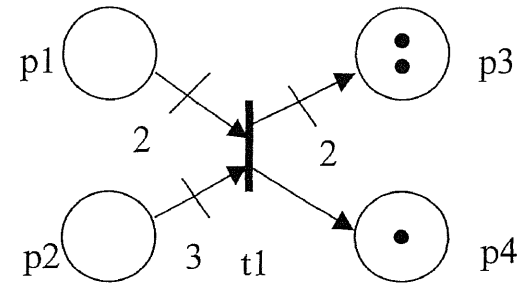
**Table 2.1** Graphical Representation of Petri Net Components

Component	Description
	Place
	Immediate transition (no firing time)
	Timed transition
	Token
	Arc illustrating input and output mapping
	Self-loop arc
	Inhibitor arc
	$n$ =Number of tokens associated with a place



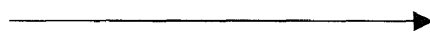


(a) t1 is enabled

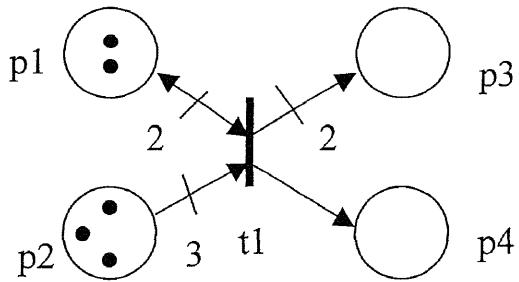


(b) t1 is not enabled

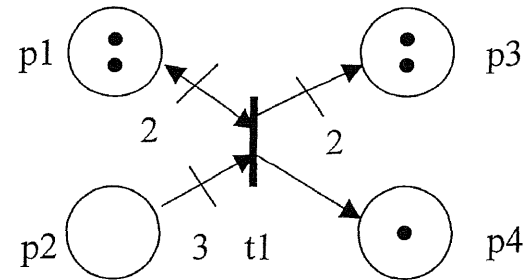
Before firing t1



After firing t1



(c) t1 is enabled



(d) t1 is not enabled

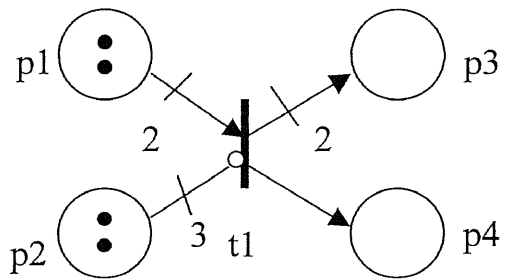
Before firing t1



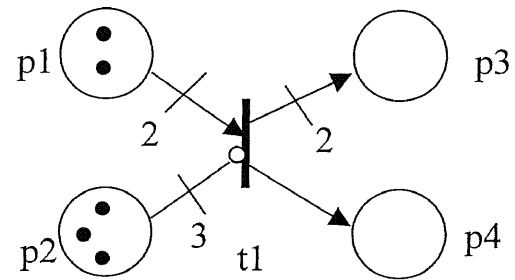
After firing t1

PN with Self-Loop Arcs

**Figure 2.4** Petri Net Execution Rules



(e) t1 is enabled



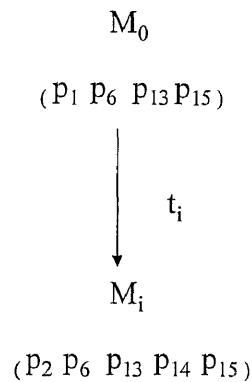
(f) t1 is not enabled

PN with Inhibitor Arcs

Figure 2.4 Petri Net Execution Rules

***Reachability:***

Given a PN  $Z=(P, T, I, O, M_0)$ , marking  $M_i$  is reachable from marking  $M_0$  if there exists a sequence of transitions firings that transforms a marking from  $M_0$  to  $M_i$ . Marking  $M_i$  is said to be immediately reachable from  $M_0$  if firing an enabled transition in  $M_0$  leads to  $M_i$ . Reachability checks whether the system can reach a particular state, or exhibit a specific functional behavior. Fig. 2.5 presents  $M_i$  each place is associated with a natural number and this number represents the number of tokens in that place at any given time. For example, if there are three places  $p_1, p_2$  and  $p_3$  then the marking  $M_i (0 \ 1 \ 2)$  represents none in place  $p_1$ , one token in place  $p_2$  and two tokens in place  $p_3$ . In Chapter 4, since the Petri Net model for the system of the CAN FieldBus access protocol has 15 places and 14 transitions, it is difficult to draw a reachability graph for a system, each of whose markings contains 15 natural numbers. To simplify this situation, in our reachability graph we indicate only those places in the marking that has tokens. For instance, initial marking  $M_0 (p_1 \ p_6 \ 2p_{13} \ p_{15})$  indicates that there is one token in places  $p_1, p_6,$  and  $p_{15}$  and two tokens in place  $p_{13}$  and all other places have zero tokens. An advantage of this notation is that we can easily identify the places that are responsible for the firing of the transition and subsequently changing the state of the system.



**Figure 2.5** Example of Marking and Firing Transition

***Boundedness and Safeness:***

Given a Petri net  $Z=(P, T, I, O, M_0)$  and its reachability set  $R$ , a place  $p \in P$  is  $B$ -bounded if  $M(p) \leq B, \forall M \in R$  where  $B$  is a positive integer, and " $\in$ " means "belong to" and " $\forall$ " means "for all".  $Z$  is  $B$ -bounded if each place in  $P$  is  $B$ -bounded. Safeness is 1-boundedness.

Places are frequently used to represent information storage areas in communication and computer systems, and product, pallet, and tool storage areas in manufacturing systems. It is important to be able to determine whether proposed control strategies avoid the overflow of these storage areas and capacity overflow of these resources [6]. The boundedness of a Petri net is used to identify the existence of overflows in the modeled system. When a place models an operation, its safeness guarantees that the controller has no attempt to initiate an ongoing process.

***Liveness:***

The concept of liveness is closely related to the deadlock case, which has been studied extensively in the context of operating systems.

A transition  $t$  is "live" when at any marking  $M \in R$ , there is a sequence of transitions whose firing reaches a marking that enables  $t$ . A Petri net is "live" if every transition in it is live.

A transition  $t$  is "dead" when there is  $M \in R$ , such that there is no sequence of transition firings to enable  $t$  starting from  $M$ . A Petri net contains "deadlock" as there is  $M \in R$  at which no transition is enabled. Such a marking is called a dead marking.

Deadlock cases occur as a result of inappropriate resource allocation policies or exhaustive utilization of some or all resources. In a resource sharing environment, the following four conditions [6] must hold for a deadlock to arise:

1. Mutual exclusion: a resource is either available or allocated to a process which has an exclusive access to this resource.
2. Hold and wait: a process is allowed to hold a resource(s) while requesting more resources.
3. No preemption: a resource(s) allocated to a process cannot be removed from the process, until it is released by the process itself.
4. Circular wait: two or more processes are arranged in a chain in which each process waits for resources held by the process next in the chain.

Liveness of a Petri net means that for any marking  $M$  reachable from the initial marking  $M_0$ , it is ultimately possible to fire any transition in the net by progressing through some firing sequence. Therefore, there is no deadlock when a Petri net is live. In

other words, a Petri net is structurally live if there is a finite initial marking which makes the net live.

### 2.3.2 Timed PNs

Since an operation needs to take a finite amount of time to complete, when time is added to the firing of the transitions, it is possible to calculate some important properties of a system such as cycle times, delay and throughput. Two types of delays are associated with a PN model: deterministic delay and stochastic delay.

#### *Deterministic Delay:*

A deterministic timed Petri net yields if either zero or positive time delays are associated with transitions. The tokens are removed from the input places immediately when the transition is enabled, but the tokens are held by the transition for a fixed period of time until they are put into the output places. Time may also be associated with places. A token deposited in a place becomes available after the time delay associated with the place.

#### *Stochastic Delay:*

Stochastic Petri nets (SPN) feature each transition with an exponentially distributed random time delay. In generalized stochastic Petri nets, there are two kinds of transitions, timed and immediate transitions. The latter has zero time delay. It is necessary to specify a discrete probability distribution function to each of immediate transitions so that if several immediate transitions have input arcs from the same place, these transitions must can be selected to fire based on their probability functions.

## CHAPTER 3

### CAN FIELDBUS

#### 3.1 FieldBus

##### 3.1.1 FieldBus Model and Specifications

FieldBus network is based on the OSI seven-layer model. The seven layers are physical layer, data layer, network layer, transport layer, session layer and application layer as shown in Fig. 3.1. The seven-layer protocol is usually too slow to be used for real-time communication. Therefore, the FieldBus uses only three layers: 1) the Physical Layer, 2) the Communication "Stack", and 3) the User Application, as shown in Figure 3.1 [1]. Because the FieldBus does not use the OSI layer 3rd, 4th, 5th, and 6th layers, the communication stack is comprised of the 2nd and 7th layers in OSI model. There is no user application in the OSI model. The FieldBus has specified a User Application model. The FieldBus reserves the required communication capacity in advance for each real-time channel (a point-to-point unidirectional connection which can provide end-to-end delivery delay guarantees), and uses a centralized scheme for scheduling messages in order to guarantee the delivery of real-time messages before their deadlines.

##### *FieldBus Communication Stack:*

Since the FieldBus communication stack includes the data link layer (DLL), the FieldBus Access Sublayer (FAS) and FieldBus Message Specification (FMS), it will complete the following operations:

- 1) The DLL controls transmission of messages onto the FieldBus. The DLL manages access to the FieldBus through a deterministic centralized bus scheduler called the Link Active Scheduler (LAS).
- 2) The FAS uses the scheduled and unscheduled features of the Data Link Layer to provide a service for the FieldBus Message Specification (FMS).
- 3) The FMS services allow user applications to send messages to each other across the FieldBus using a standard set of message formats. The FMS describes the communication services, message formats, and protocol behavior needed to build messages for a user application.

The FieldBus Access Sublayer (FAS) maps the FMS onto the DLL. Fig. 3.2 shows the frame at the data link layer of the FieldBus network. Each layer in the communication system is responsible for a portion of the message that is transmitted on the FieldBus [1]. The preamble is used by a receiver to synchronize its internal clock with the incoming FieldBus signal. The preamble consists of a minimum of 8 zeros (i.e., transitions) that allows the digital phase lock loop (DPLL) to adjust its frequency until it is synchronized with the transmitter clock. Since frame synchronization involves identifying the beginning and end of frames, and also ensuring its error-free transmission, the synchronization is performed by having flag fields as the start delimiter and the end delimiter in the frame format, as shown in Fig. 3.2, along with the frame check sequence (FCS) field. The flag fields have the unique bit pattern 01111110. A data link protocol should be transparent, in that it should allow the transmission of any bit pattern. Content synchronization is achieved by having the frame structure which allows for the distinction of control and data frames and the exchange of I-frames (information frames)



and S-frames (supervisory frames). Content synchronization using I-frames involves checking the sequence numbers which are used to perform flow control purposes. Dialog synchronization is achieved by exchanging U-frames (unnumbered frames).

The FieldBus has defined a standard User Application based on "Blocks". Blocks represent the different types of application functions such as resource block, function block and transducer block. They are described below.

- 1) The resource block describes characteristics of the FieldBus device. There is only one resource block in a device;
- 2) The function blocks provide the control system behavior. The input and output parameters of function blocks can be linked over the FieldBus. There can be many function blocks in a single User Application;
- 3) The transducer blocks decouple function blocks from the local input/output functions required to read sensors and command output hardware. There is usually one transducer block for each input or output function block.

The Data Link Layer service of FieldBus provides both connectionless and connection-oriented communication between two peer communicating data link entity (DLE). The connection-oriented communication service supports both real-time and nonreal-time communication. The connectionless service supports only nonreal-time communication. There are three classes of DLE's in the FieldBus data link layer protocol: Basic, Link Master (LM), and Bridge. The LM class DLE's can also act as link active scheduler (LAS), but each time exactly one LM DLE can act as the LAS of a FieldBus link. For each FieldBus link, FieldBus has a control unit, LAS DLE. The LAS DLE is responsible for scheduling messages on the local link. A bridge DLE performs a store and

forward function to connect two or more separate multiaccess links. The link's LAS schedules all the normal communication requests for the use of a link. Since most of real-time communication is likely to take place between two peer DLE's on the same link, most of time-critical communication can be handled by the local LAS.

In Fig. 3.2, the FieldBus Physical Layer receives messages from the communication stack and converts the messages into physical signals on the FieldBus transmission medium. The numbers show the number of eight bit "octets" used for each layer to transfer the user data.

### **3.1.2 Two Speed FieldBuses**

FieldBus has high performance because it moves data fast. Both traditional point-to-point and multidrop (share-bus) connections may be combined by the FieldBus approach. Because the instruments controlled by analog 4—20-mA current loops will be around for a long time, the Foundation FieldBus accommodates traditional and digital instrumentations. The topology of a FieldBus system is made up of two main parts [2]: a lower-speed FieldBus, often called H1 (31.25kb/s), runs 10-20 times faster than today's hybrid protocols, and a higher-speed one called H2 (may operate at either of two speeds: 1 or 2.5Mb/s). The 31.25 Kbit/s FieldBus can be used for control applications such as temperature, level and flow control. The 1.0 Mbit/s and 2.5 Mbit/s FieldBus are typically used for advanced process control, remote input/output and high speed factory automation applications. There may be three categories for FieldBus devices in use today which are the traditional analog and discrete I/O; the hybrid analog and digital devices; and purely digital devices. The tradition instruments include pump, valve, and

transmitter; hybrid instruments include local (plant floor) display, transmitter, valve, and handheld configurator; and digital instruments include transmitter, valve, process analyzer and pump. For example, a pump may be asked about its own internal temperature, which will help the plant engineers schedule preventive maintenance activities. Similarly, a valve may be asked to report its own status, e.g. how many turns of the stem it is from being completely open or closed, and a temperature transmitter may be asked when it was last calibrated. A bridge is used to interconnect different FieldBuses in speed (Figure 3.3) [2].

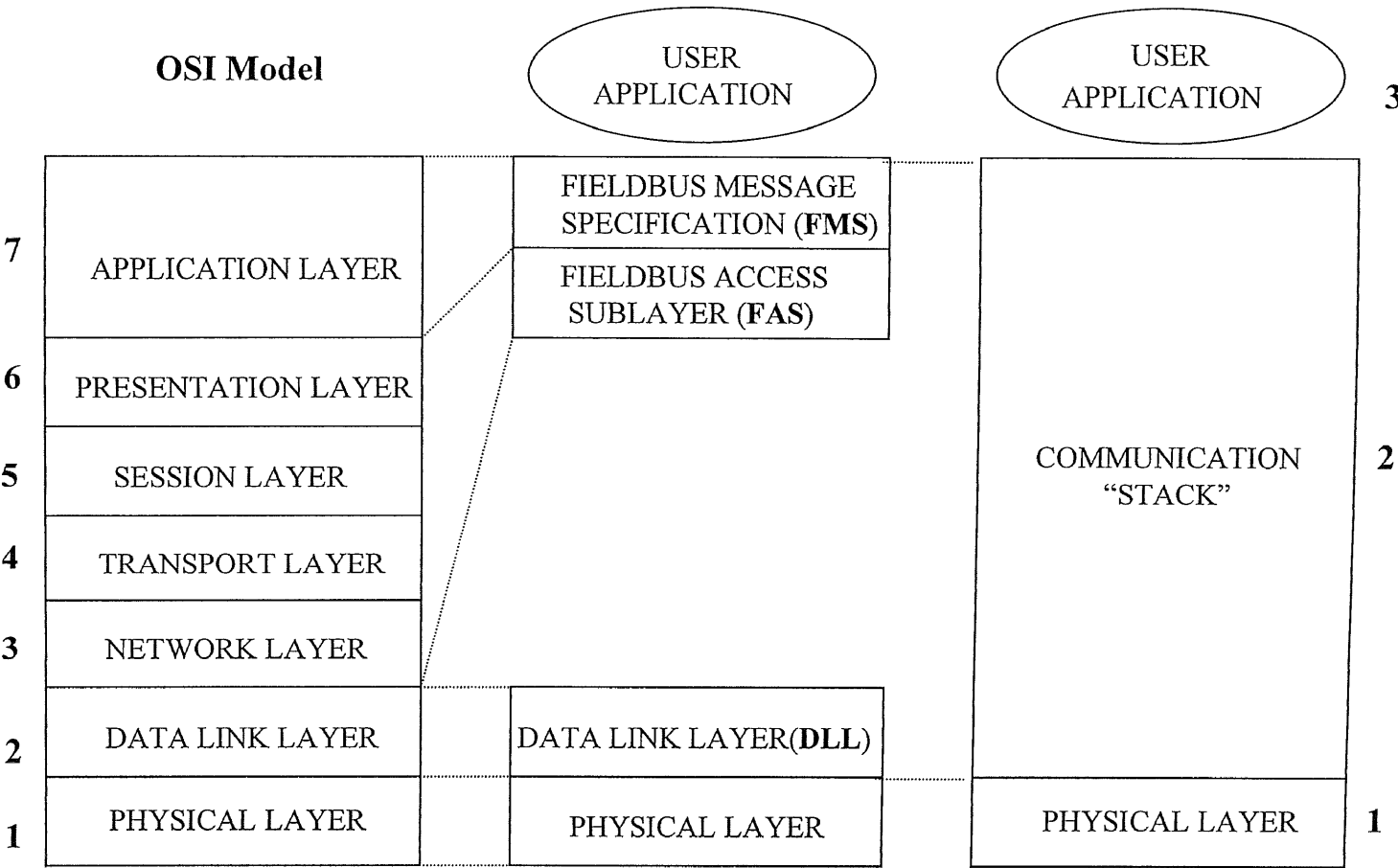
FieldBus supports both intracell and intercell real-time communications [10]. The capacity of each link is divided into intralink and interlink. LAS (the local link active scheduler) manages intracell (or: intralink) communication. A global network manager manages intercell (or: interlink) communication. In this way, it allows for fast local intracell connection establishment, while supporting global intercell connections. The LAS of a link can reserve a different fraction of link capacity for local (or intra-link) communication demands. Intralink (intracell) communications occur between two peer DEL's on the same multiaccess link. This type of communication can be either connection-oriented or connectionless. Since real-time communication requires a bounded delay in message delivery, all the required resources are reserved in advance in order to guarantee all real-time messages to be delivered before their deadlines. If there are already too many real-time connections established nearly exhausting all the link capacity, the LAS can request more link capacity to the global network manager for local usage. To interlink communication, the entire operation for establishing a connection between two DLE's on different links is similar to the case when they are on the same

link. The only difference is that the real-time connection requests are granted by the network manager, rather than a local LAS.

### 3.2 CAN Model

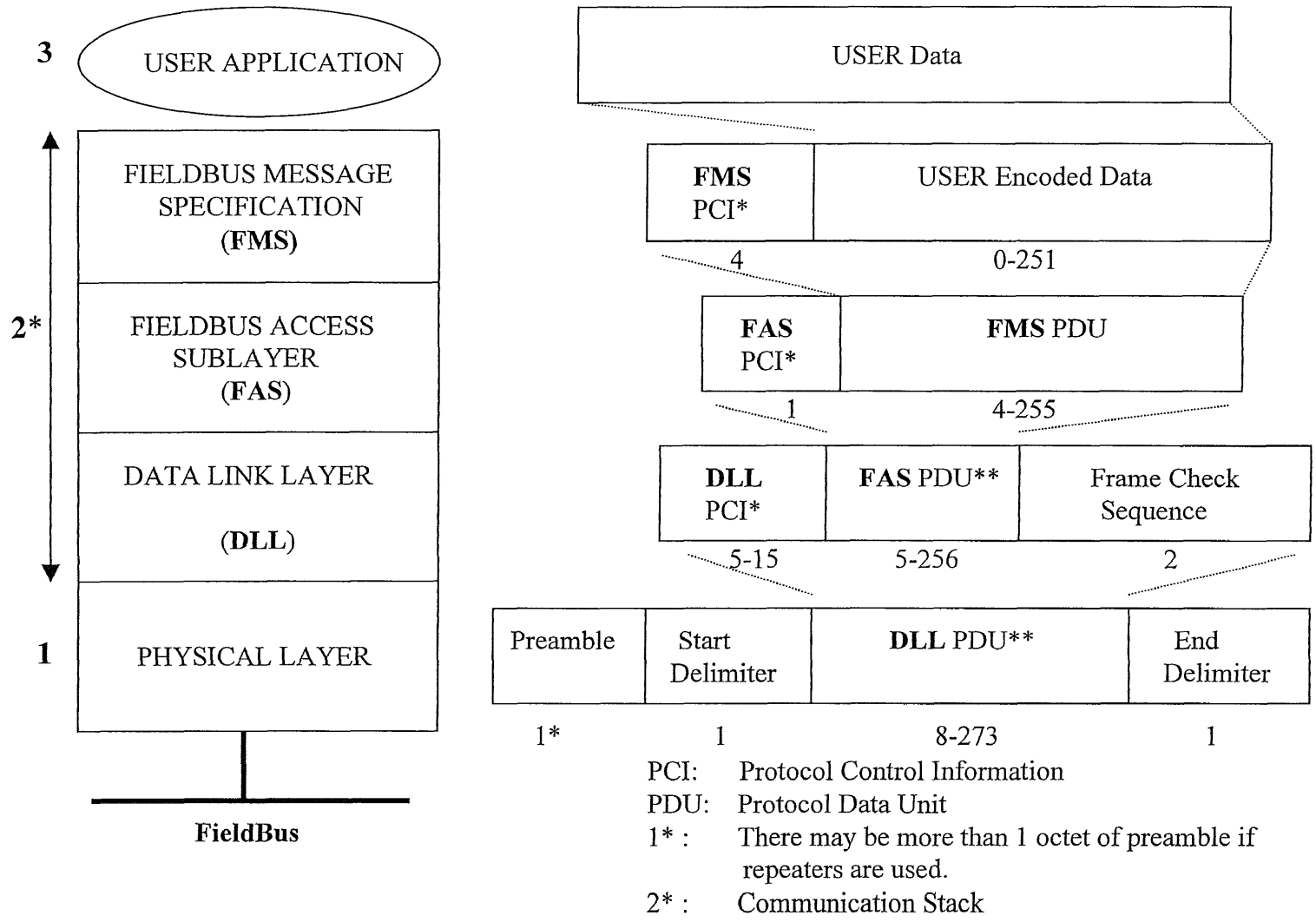
- *CAN User Application Layer*: Many applications of CAN require services that are beyond the basic functionality specified by the Data-Link Layer, but which may be implemented at the User Application Layer.
- *CAN Data-Link layer*: CAN data-link layer is the only layer that recognizes and understands the format of messages. This layer constructs the messages to be sent to the Physical Layer, and decodes messages received from the Physical Layer. In CAN controllers, the Data-Link Layer is usually implemented in hardware. Because of its complexity, the Data-Link Layer is subdivided into Logical Link Control (LLC) Layer and Media Access Control (MAC) Layer.
  1. The LLC Layer manages transmission and reception of data messages to/from other higher level layers in the model.
  2. The MAC Layer encodes and serializes messages for transmission, and decodes received messages. It also manages prioritization (arbitration), error detection, and access to the Physical Layer.
- *CAN Physical Layer*: This layer specifies the physical and electrical characteristics of the bus, and of the hardware that converts the characters of a message into electrical signals for transmitted messages. The Physical Layer is always "real" hardware.

**FieldBus Model**



Note: The user application is not defined by the OSI Model

**Figure 3.1** OSI and FieldBus Models



**Figure 3.2** Frame Format at the Data Link Layer of the FieldBus Network

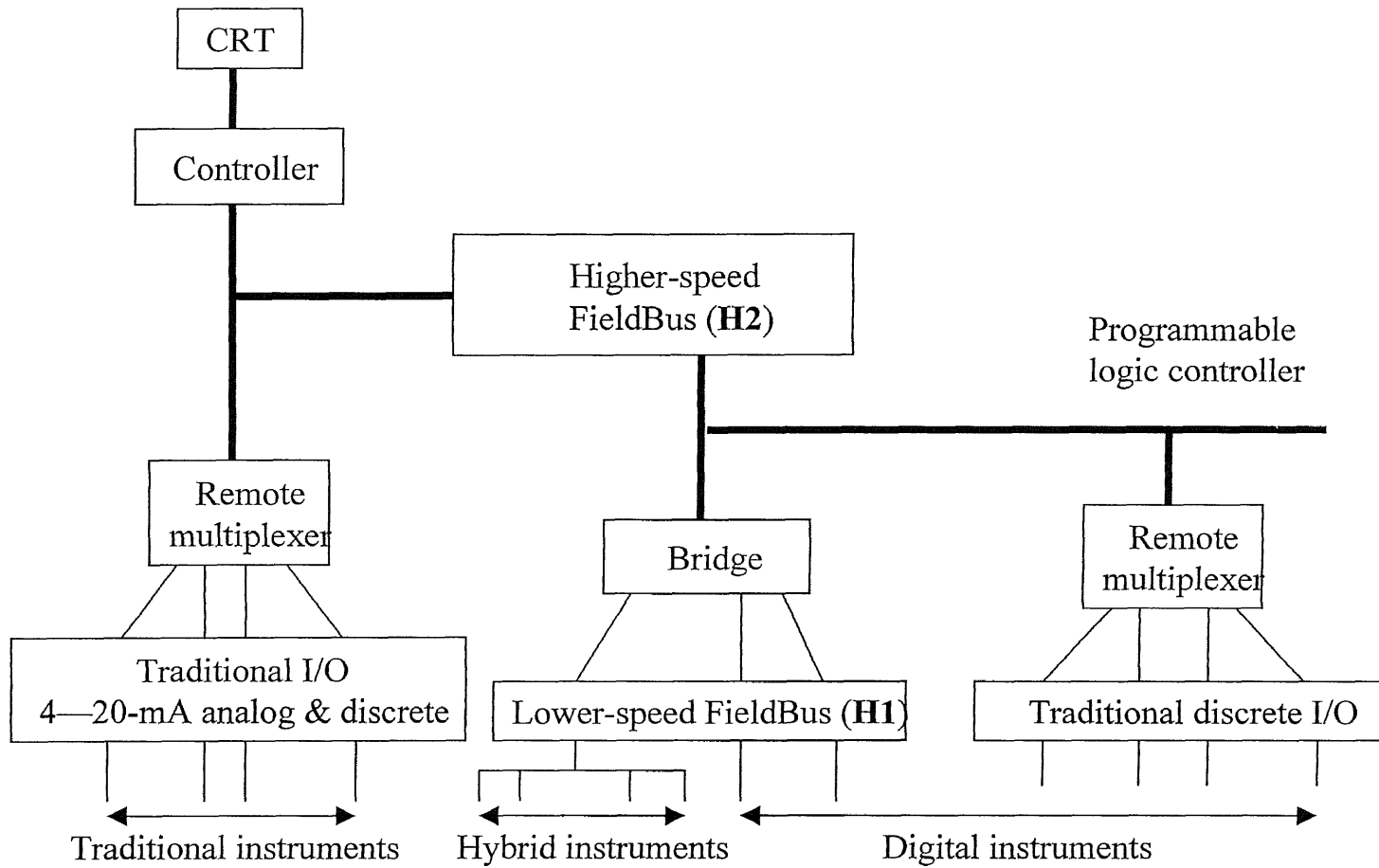


Figure 3.3 Lower and Higher Speed FieldBuses

## CHAPTER 4

### MODELING AND ANALYSIS BY USING PETRI NETS

#### 4.1 Modeling CAN FieldBus with Petri Nets

To evaluate the nature of the CAN FieldBus access protocol, it is essential to model various services and operations for the source and destination of the system by using Petri nets. A sequence of events can be known in a better fashion from the graphical representation of the model. We developed Petri nets for initialization (channel access) phase, processing (message transmission) phase, and termination (channel release) phase for the source and destination. The states such as idle, ready to send frame and events such as "send frame" and "receive frame", form the places and the transitions of the Petri net model. When a certain event takes place, the source or the destination changes its state to a new one and corresponding signals are sent to the source station or the destination station or to the network. The following section describes the PN model for the entire system of the CAN FiledBus access protocol.

##### 4.1.1 Modeling Methodology

- *Places model are often used to model:*
  1. Input conditions and output conditions;
  2. States of objects/transferring medium (e.g., "busy", "sending" and "idle"); or
  3. Counter sizes.
- *Transitions model are often used to represent:*
  1. Message transfer;



2. Interrupts;
3. Processing activity;
4. Send event; or
5. Receive event.

Modeling methodology is key to the application of Petri nets to the CAN FieldBus system. We adopt the following approach to modeling our FieldBus protocol based on the method in [7].

1. Identify the activities and resources required for the transmission and reception of a frame or message.
2. Order activities by the precedence relations as given in the process plans.
3. For each activity in order: create and label a place to represent the status of that activity; add a transition (start activity) with an output arc(s) to the place(s); add a transition (stop activity) with an input arc(s) from the activity place(s). In general, the stop transition for one activity will be the same as the start transition for the next activity. When the net is executed, a token in an activity place will indicate the activity is taking place. Multiple tokens will indicate the activity occurring in multiplicity, for example, in a counter place, two tokens might represent two frames being stored at the same time. The firing of the start transition represents starting the activity and the firing of the stop transition represents the completion of the activity and may also represent the start of the next activity.
4. For each activity in order: If such a place has not been already created, create and label a place for each resource which must be available to start the activity. Connect all appropriate resource availability places with arcs such that each inputs to the

starting transition for the activity. Create output arcs to connect the stop transition following the activity to any resource places representing resources which become available (are released) upon completion of the activity. Also create some places which serve for control purposes and link them to those transitions to implement certain control strategy.

5. Specify the initial marking for the system.

#### 4.1.2 The Modeling of Four Primitives and Concepts with Petri Nets [8]

- *Synchronization*: The synchronization between two conditions can be realized through two places linking to a common transition (e.g., from  $p_3$  and  $p_9$  to  $t_3$  in Fig. 4.1). If only one place  $p_3$  has a token, the transition  $t_3$  has to wait for another place  $p_9$  to get a token. Then the transition can be fired. Therefore, a transition in a net may have multiple input places, as shown in Fig. 4.2(a).
- *Choice*: In the context of Petri nets, a place has two or more outgoing arcs, representing the different possible events to follow as shown in Fig. 4.2(b). For example, in the Petri net for the CAN FieldBus, place  $p_3$  ( $p_8$ ) has two outgoing arcs to transitions  $t_3$  ( $t_{10}$ ) and  $t_7$  ( $t_{11}$ ) in Fig. 4.1. In two arcs, one must be chosen probabilistically.
- *Fork*: The fork structure is used to model an event whose occurrence leads to more than one information and/or control signal, as shown in Fig. 4.2 (c). For example, consider transition  $t_{11}$  in Fig. 4.1. When the frame processed fails, the system sends out a negative response to the source, returns one token to the counter place, and the destination becomes available.

- *Join/Merge*: A place may have more than one incoming arc, signifying that several information/control sources lead to a common condition/status represented by a place, as shown in Fig. 4.2 (d). For example, in Fig. 4.1, depositing a token in place  $p_2$  means sending frame is available. Release from any process (e.g., from transitions  $t_1$ ,  $t_4$ ,  $t_7$  or  $t_{12}$  to place  $p_2$ ) resumes the availability of the place  $p_2$ .

#### 4.1.3 A Modeling Example for the System

To illustrate the modeling methodology in a more general way, take some samples in the system which can be modeled with places, transitions and arcs. The system consists of source and destination stations, and test and control parts. The following is the modeling methodology for the system:

1. The activities required are sending and receiving messages, sending and receiving positive or negative response, testing the result of the sent frame, and the resources are two stations (or source and destination stations), a counter, and messages.
2. The order of activities is as follows:
  - *Source Station*
    - a. Establish a channel connection;
    - b. Send a frame;
    - c. Receive a positive or negative response;
    - d. Check for more frame or send a frame again;
    - e. Send the frame again or Disconnect a channel.
  - *Destination Station*
    - a. Receive a frame;

- b. Process a frame;
  - c. Send a positive or negative response.
3. For each activity in order:

- *Source Station:*

As shown in Fig. 4.1 and Table 4.1, places  $p_2$ ,  $p_3$ ,  $p_4$ ,  $p_5$  and  $p_7$  are created to model the activity sequence for the part of the source station. Transition  $t_1$  models the start of the activity of establishing a channel;  $t_2$  models the stop of the activity of establishing a channel and the start of sending a frame and self-test;  $t_3$  ( $t_7$ ), the end of sending a frame and the start of receiving a positive response (negative response);  $t_4$ , the end of receiving a positive response and the start of sending a next frame again;  $t_5$ , the end of receiving a positive response and the start of disconnection;  $t_6$ , the end of disconnection and the start of establishing a next channel.

- *Destination Station:*

The places  $p_7$ ,  $p_8$ ,  $p_9$ , and  $p_{10}$  are created to model the activity sequence for the part of the destination station. Transition  $t_8$  models the start of the activity of receiving a frame;  $t_9$ , the end of receiving the frame and the start of processing the frame;  $t_{10}$  ( $t_{11}$ ), the end of processing the frame and the start of sending a positive response (a negative response).

4. For activity  $p_2$  (ready to send a frame) we require that an initialization request be available. Fig. 4.1 shows the Petri net model with place  $p_1$  representing an initialization request available. It has an input arc to transition  $t_1$ . Next consider  $t_2$  as the end of the activity of establishing a channel and the start of the activities of sending a frame and self-test. To start these activities, counter( $p_{13}$ ) and operation control ( $p_{15}$ ) must be available. These are modeled by an input arc from  $p_{13}$  to  $t_2$ ,

another input arc from  $p_{15}$  to  $t_2$ , and the output arcs from  $t_2$  to  $p_{11}$  for self-test and from  $t_2$  to  $p_3$  for waiting for response. To the counter activity,  $p_{13}$ , in order to send a frame, empty spaces in the counter slot must exist. This is modeled by place  $p_{13}$  where the number of tokens indicates the number of empty spaces (or this number would be the number of frames to be sent later) and arc from  $p_{13}$  to  $t_2$  the start transition of  $p_3$ . The completion of the counter activity for sending frames and the release of empty spaces is represented by transition  $t_{14}$  and  $t_{11}$ , the arcs from  $t_{14}$  to  $p_{13}$  and from  $t_{11}$  to  $p_{13}$ . Other activities and resources are modeled in the similar ways.

5. The initial marking is formulated for system startup. The message, the counter, the receiver and control places should be available to start. The initial marking for these conditions is shown in Figure 4.1. Note that in this initial marking only transition  $t_1$  is enabled, implying a frame is ready to be sent out.

Figure 4.1 shows the communication behavior between the source station and the destination station during three phases (channel access, message transmission, and channel release phases).

- *Communication Process in the Source Station:*

Suppose each message has "k" frames (counter = k in Fig. 4.1). Initially the source is in an "idle" state, or "initialization request" state, represented by place  $p_1$ . Upon receiving an initialization request, transition  $t_1$  fires. The firing of transition  $t_1$  indicates the incoming initialization request which means to set up a channel connection for sending a message. Once a channel connection is established, the source is ready to send a frame. At this point a token is deposited into the output place  $p_2$ . The tokens in  $p_2$ ,  $p_{13}$ , and  $p_{15}$  enable transition  $t_2$  and its firing deposits the tokens in the output places  $p_{11}$  (self-test) and  $p_3$

(frame sent and waiting for response). The source might receive different responses (positive or negative response) regarding the results of the frame processed. As long as there are one or more tokens in the counter place  $p_{13}$ , the frame can be issued from the source. When the result of the "self-test" in the place  $p_{11}$  fails, the tokens in  $p_{11}$  and  $p_3$  enable transition  $t_{12}$  and its firing deposits the tokens in the output places  $p_2$ ,  $p_{15}$  and  $p_{13}$ . That means: 1) a token is returned to counter place  $p_{13}$ ; 2) the original frame needs to be sent again; 3) a control token is sent back to  $p_{15}$ . If the result of the "self-test" succeeds, the token in  $p_{11}$  enables transition  $t_{13}$  and its firing deposits the token in the output place  $p_{12}$  (frame sent by sender).

- *Communication Process in the Destination Station:*

When there is a token in place  $p_{12}$ , the tokens in  $p_{12}$  and  $p_6$  enable transition  $t_8$  and its firing deposits the token in the output place  $p_7$  (frame received). At this point, the frame is sent to the destination, Then, the token in  $p_7$  enables transition  $t_9$  and its firing deposits the token in the output place  $p_8$  (frame processed). The destination changes its state to frame processed after receiving the frame. Depending upon the nature of the destination response (positive or negative response after the frame processed), transition  $t_3$  or  $t_7$  in the source station fires and the system changes its state to "ready to send next frame" or "ready to disconnect". When the frame processed by the destination fails, the token in  $p_8$  enables transition  $t_{11}$  and its firing deposits the tokens into  $p_{10}$  (negative response sent),  $p_6$  (ready to receive) and  $p_{13}$  (returning a counter slot), or the destination sends the negative corresponding response to the source. After the source receives the negative response, the tokens in  $p_{10}$  and  $p_3$  enable transition  $t_7$ , the tokens go back to  $p_2$  (ready to send frame again) and  $p_{15}$  (control token is reset). When the frame processed by the destination

succeeds, the token in  $p_8$  enables transition  $t_{10}$  and its firing deposits the tokens into the output places  $p_6$  (ready to receive),  $p_9$  (positive response sent) and  $p_{14}$  (counter). The source starts receiving the positive response, and the tokens in  $p_9$  and  $p_3$  enable transition  $t_3$  and its firing deposits the token in output place  $p_4$  (checked for more frame).

- *Counter Operation Process:*

In order to keep track of the number of the sent frames of a message, a counter is set in the PN model. This counter (place  $p_{14}$ ) is used as the source frame counter. Once a frame is completed successfully, the following frame request increments the counter. In this mode transition  $t_3$  fires if the positive response is sent out from the destination or the transition  $t_7$  fires if the negative response is sent out from the destination. As long as there are one or more frames in progress, the tokens in  $p_4$  and  $p_{13}$  enable transition  $t_4$  and its firing deposits the tokens into  $p_2$ ,  $p_{15}$  and  $p_{13}$ . A self-loop arc connecting the place  $p_{13}$  and transition  $t_4$  helps to check if there are more frames or not. When the transition  $t_4$  fires,  $p_{13}$  always keeps the same token number. At this point the source and the destination return to "ready to send subsequent frames" and "ready to receive" states. When there are no more tokens in  $p_{13}$ , that means no more frames to be sent, in this case "k" tokens in  $p_{14}$  enable transition  $t_5$  and its firing deposits the tokens into the output places  $p_5$  (ready to disconnect),  $p_{15}$  (control token is reset), and  $p_{14}$  ("k" tokens), then the tokens in  $p_{15}$  (control place) and  $p_{14}$  (counter) enable transition  $t_{14}$  and the "k" tokens in  $p_{14}$  all return to the counter place  $p_{13}$ .

- *Final Process:*

When there is no more frame to be sent for a given message, the token in  $p_5$  enables transition  $t_6$  and its firing deposits the token back into the idle place  $p_1$ . After the end of

the disconnection process, the source and destination go back to their original states by  $p_1$  and  $p_6$  respectively. This completes the communication process for a message.

Tables 4.1 and 4.2 present detailed description of various places and transitions for the source and destination and different signals sent during the process in the PN model. In Table 4.2 one unit time is defined as the following from the highest speed and the shortest frame:

highest speed:  $H_2=2.5\text{Mb/s}$

shortest frame:  $64\text{bits/frame}$

Then, the time for sending a shortest frame= $64*10^{-6}/2.5=25.6\mu\text{s/frame}$

One unit time= $25.6\mu\text{s}/10=2.56\mu\text{s}$

This concludes the discussion of PN modeling for various services and operations. We now focus on performance analysis of the CAN FieldBus based on the Petri Net model.

## **4.2 Performance Analysis of CAN FieldBus**

### **4.2.1 Petri Net Properties**

There are several properties of a Petri net model. The properties are mainly classified as the behavioral and structural properties. The behavioral properties depend on the initial state or marking of a Petri net. On the other hand, the structural properties depend on the net structure of a Petri net. There are three examples of the behavioral properties: reachability, boundedness, and liveness. We will describe the reachability property of a Petri net model in detail in the following section.



### 4.2.2 Reachability and Petri Net

The reachability property is very useful in checking whether a system can reach a specific state, or exhibit particular functional behavior. A reachability graph is also used to represent the set of all reachable markings. It describes the sequence of transitions that may fire in order to convert the system from one state to another.

- *Marking*: The marking of a Petri net at any case in time depicts the position of the tokens with respect to the places of the Petri net. As mentioned in Chapter 2, an initial marking ( $M_0$ ) represents the initial state of the system. The sequence of firing of transitions moves the Petri net into a new marking ( $M_i$ ).  $M_i$  describes a new state that is reached as the result of a specific sequence of transition firings.
- *Reachability Graph for the System of CAN FieldBus*: Two reachability graphs/trees are drawn for the entire system of the CAN FieldBus access protocol in the cases of one and two frames in a message ( $k=1$  and  $2$ ) in place  $p_{13}$ , respectively (Figs. 4.3, 4.4 and 4.6).

To save space, we denote a marking by just listing the marked places. In Figure 4.1 ( $k=1$ ) transition  $t_1$  is enabled in marking  $M_0$  ( $p_1$   $p_6$   $p_{13}$   $p_{15}$ ) and its firing converts the system state to  $M_1$  ( $p_2$   $p_6$   $p_{13}$   $p_{15}$ ). In  $M_1$ , transition  $t_2$  is enabled whose firing will generate a new marking  $M_2$  and so on. In a given state, there can be more than one transition enabled. We use a reachability graph to represent the behavior of a system. Depending upon the system response, the transition  $t_{10}$  or  $t_{11}$  can fire, then changing the system state to either  $M_6$  or  $M_6'$  respectively. From  $M_6'$ , since the response fails, the system goes back to the ready state to send frame and from  $M_6$  the subsequent transitions fire leading the system back to the original (or idle) state.

The reachability graph for the entire system of the CAN FieldBus access protocol is essential to integrate the different events and operations with the initialization, processing and termination phases of the system in order to compute the time required to complete these processes.

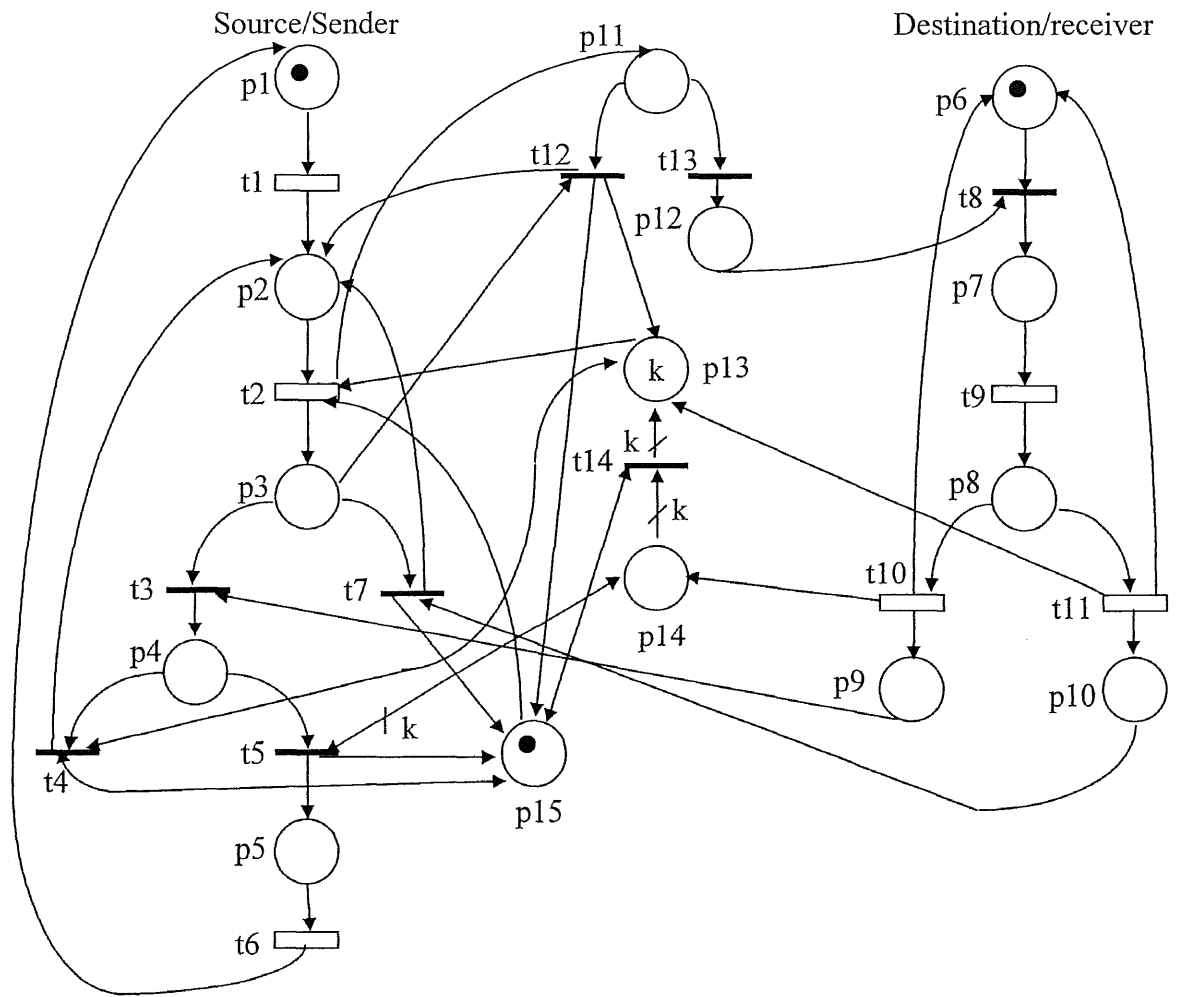
This reachability graph has two components  $M_i$  and  $t_i$  where  $M_i$  is the marking and  $t_i$  is the enabled and fired transition. We have presented this graph in Fig. 4.3. In this CAN FieldBus access protocol, there are mainly three distinct communication phases between a source station and a target: channel access, message transmission, and channel release phases. If the message transmission fails, the system will return the message to the transmission phase. If it succeeds, the system goes to the channel release phase. Based on these reachability graphs, we can conclude that the model is bounded and live given the initial markings as shown in Figs. 4.3 and 4.5 a).

### 4.3 The System Process Time Computation

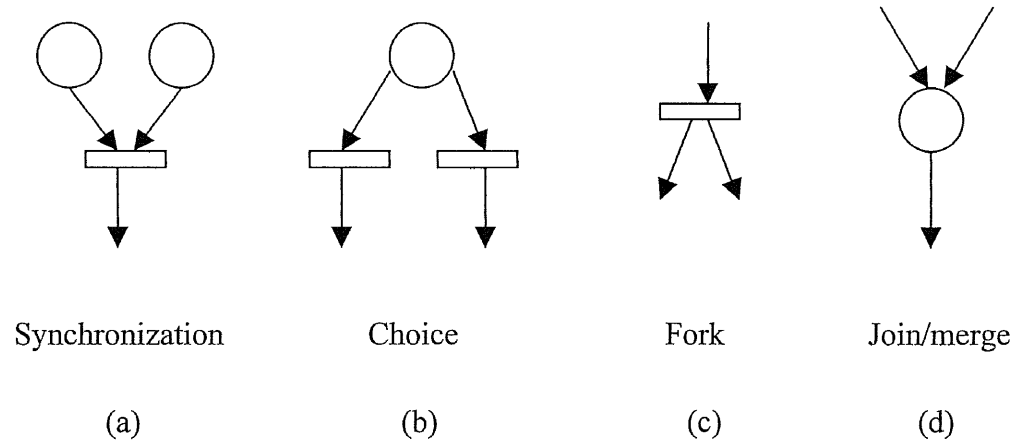
It is possible to compute the completion time for successful events and operations from the reachability graph. Different operation processes may occur for successful events and operations. The processes are indicated by loop numbers. Table 4.4 shows the time required to complete the process, and cycle time associated with different loops.

The process analysis has been carried out by assuming the delays associated with transitions as shown in Table 4.2. Table 4.3 shows the varying delay time associated with  $t_2$  (send frame). From Table 4.4 it is observed that the time required for successful events in loop<sub>3</sub> (loop<sub>4</sub>) is 22 units. It is calculated based on  $t_1+t_2+t_{13}+t_8+t_9+t_{10}+t_3+t_5+t_6(t_{14})+t_{14}(t_6)$  where  $t_i$  represents for convenience the delay associated with transition  $t_i$ .

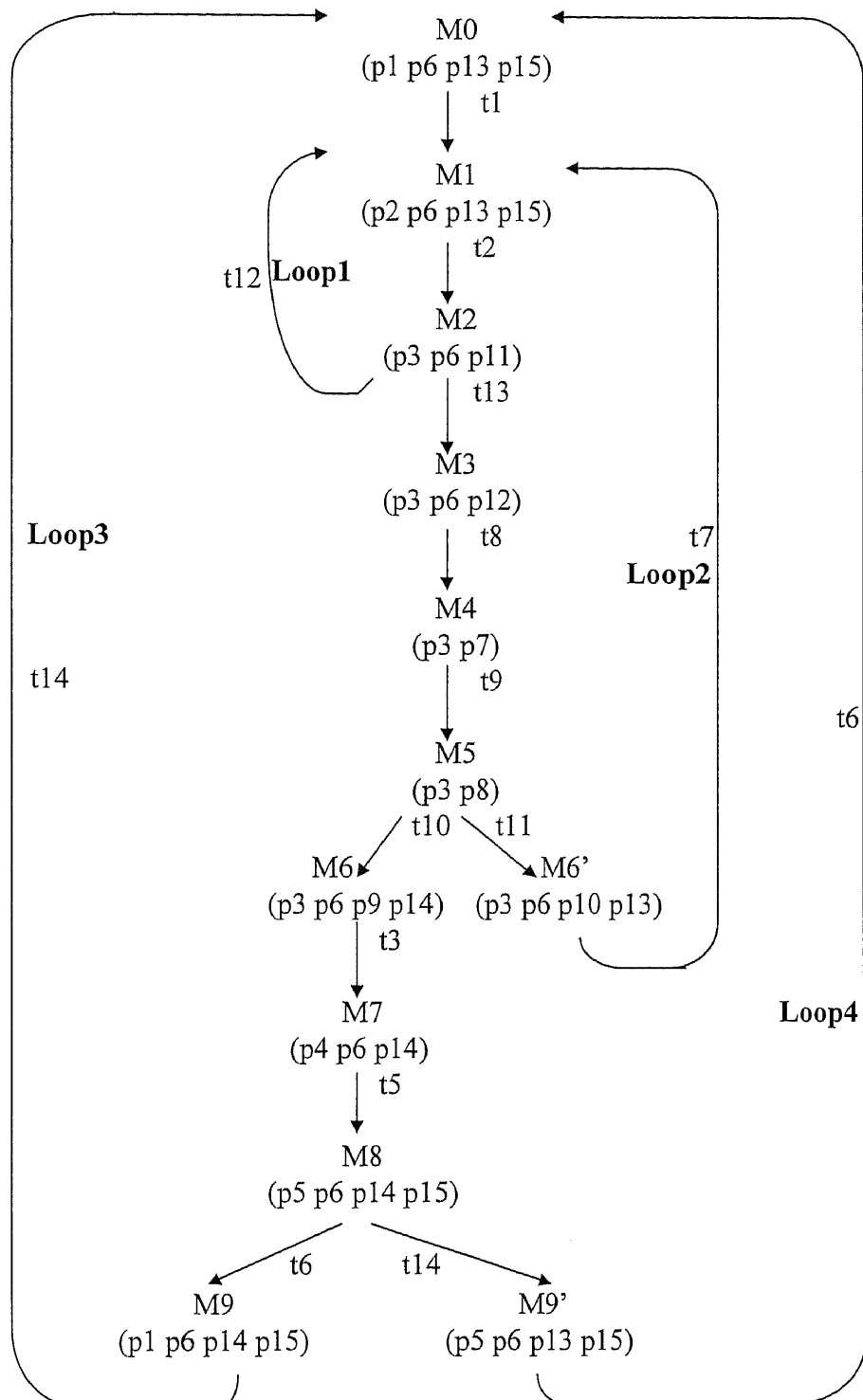
From Table 4.4, when the FieldBus speed is 2.5Mb/s and the length of a frame is 64bits, the minimum cycle time is 22 time units (or case 1), as determined by the loop<sub>3(4)</sub>. Here for simplicity, we have assumed only one frame in each message (or  $k=1$ ). This means that it takes a minimum of 22 time units to complete the communication process for one message between the source and the destination. In reality, there can be a number of frames in each message so that the reachability graph of a system becomes too complex to represent and analyze. In case 2, when the loop<sub>1</sub> (or self-test is failed) is in effect once during processing a successful communication, it will take 10 more units time (or  $22+10=32$  units). Hence the total time for this communication is increased by 45%, as shown in the Fig. 4.7. In case 3, when the loop<sub>2</sub> (or negative response) is in effect once during processing a successful communication, it will take 21 more units time (or  $22+21=43$  units). The overall time is increased by 95%. Fig. 4.7 shows the effect of the loop<sub>2</sub> on the successful operation.



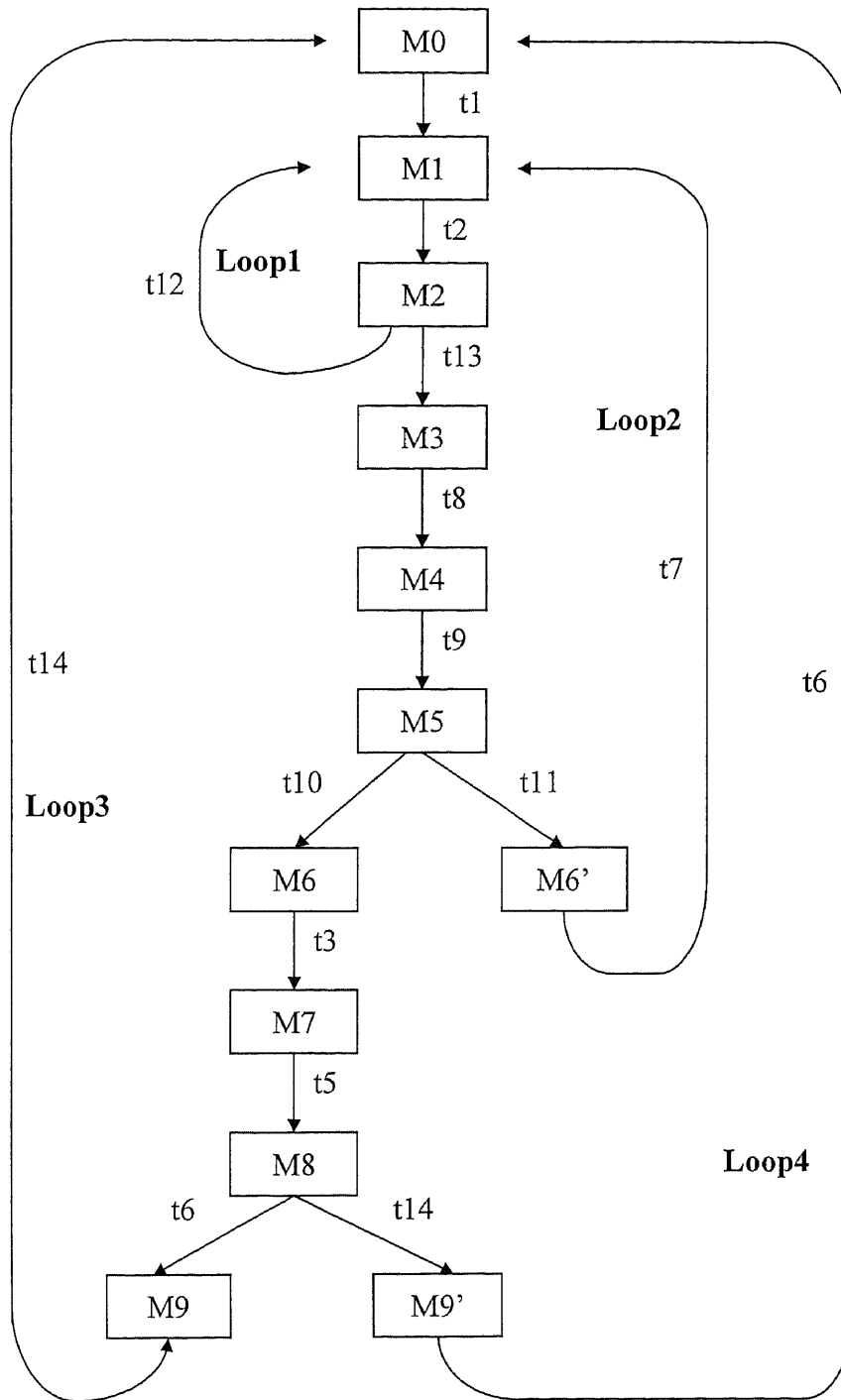
**Figure 4.1** The Petri Net Model for the CAN FieldBus Access Protocol (Counter=k)



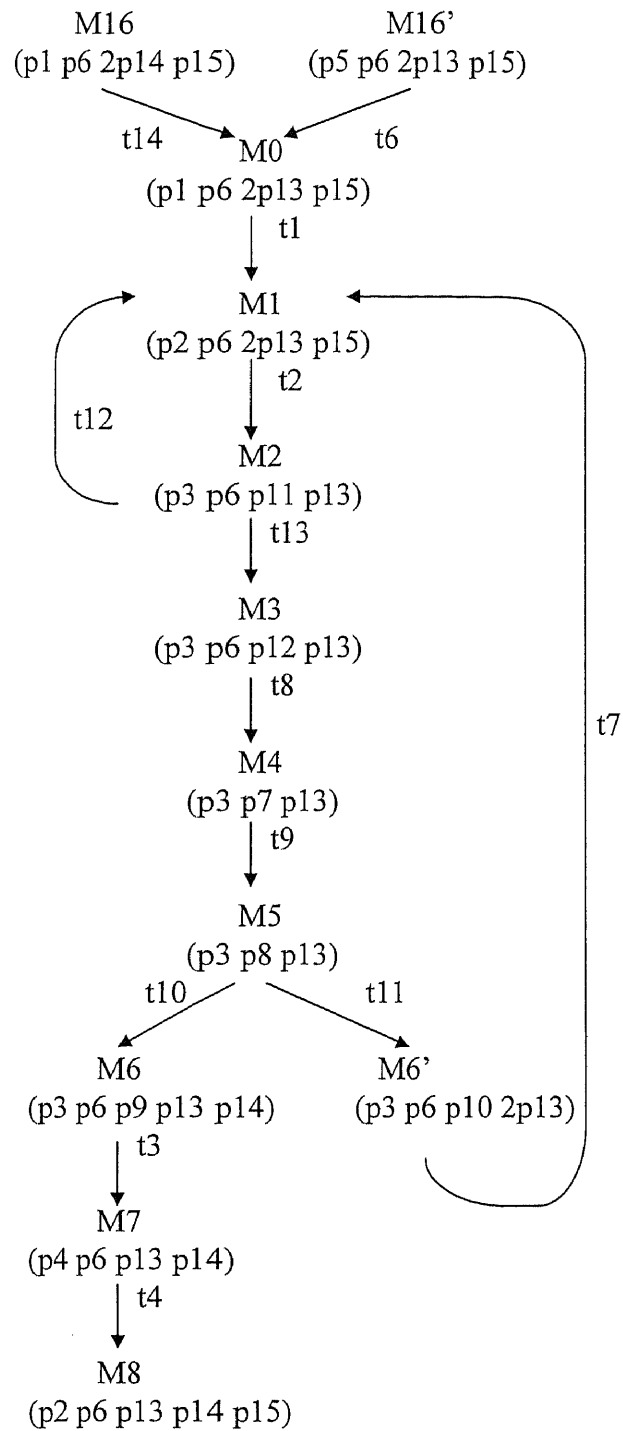
**Figure 4.2** Representation of Four Primitives



**Figure 4.3** Reachability Graph for the CAN FieldBus Access Protocol ( $k=1$ )  
 ( with only marked places shown in each marking)

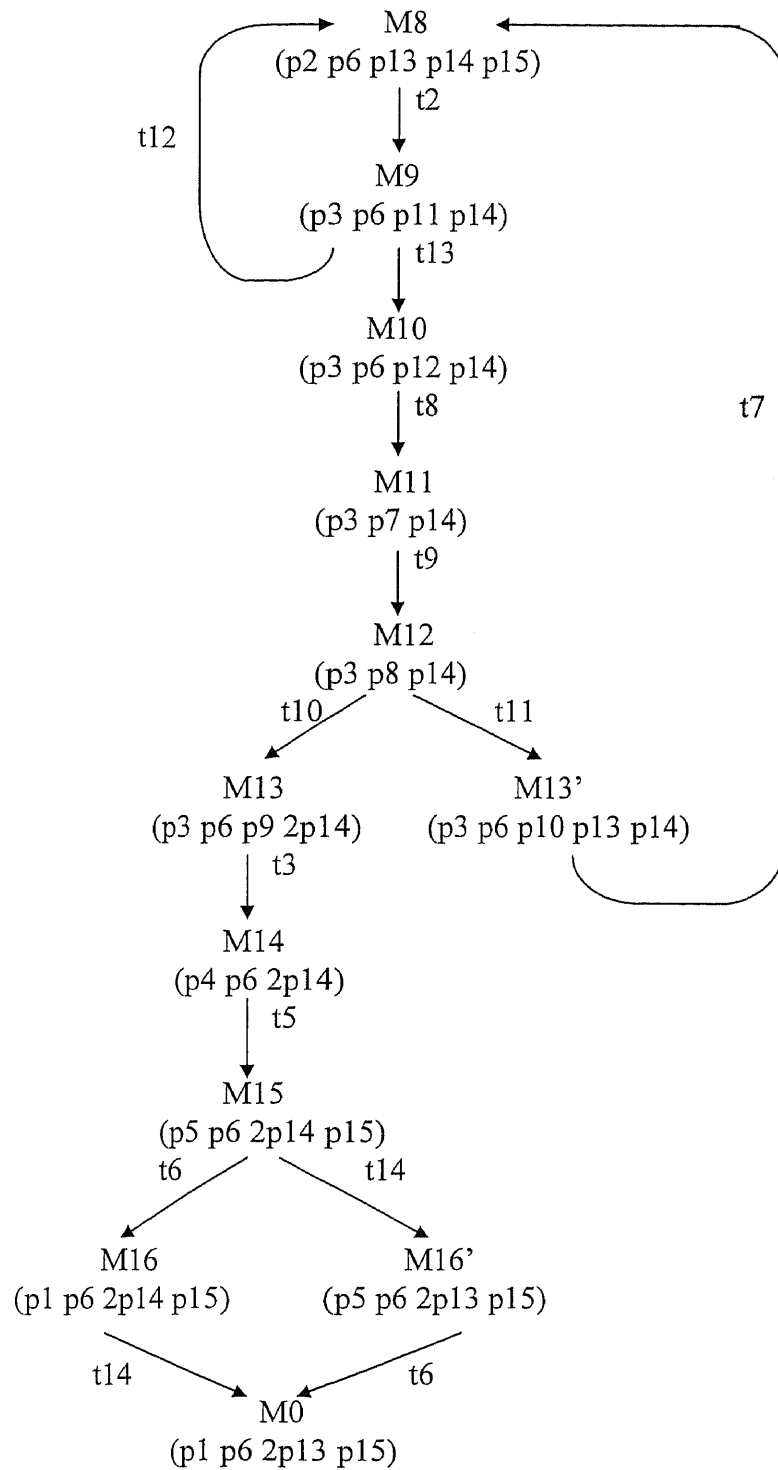


**Figure 4.4** Reachability Graph for the CAN FieldBus Access Protocol ( $k=1$ )

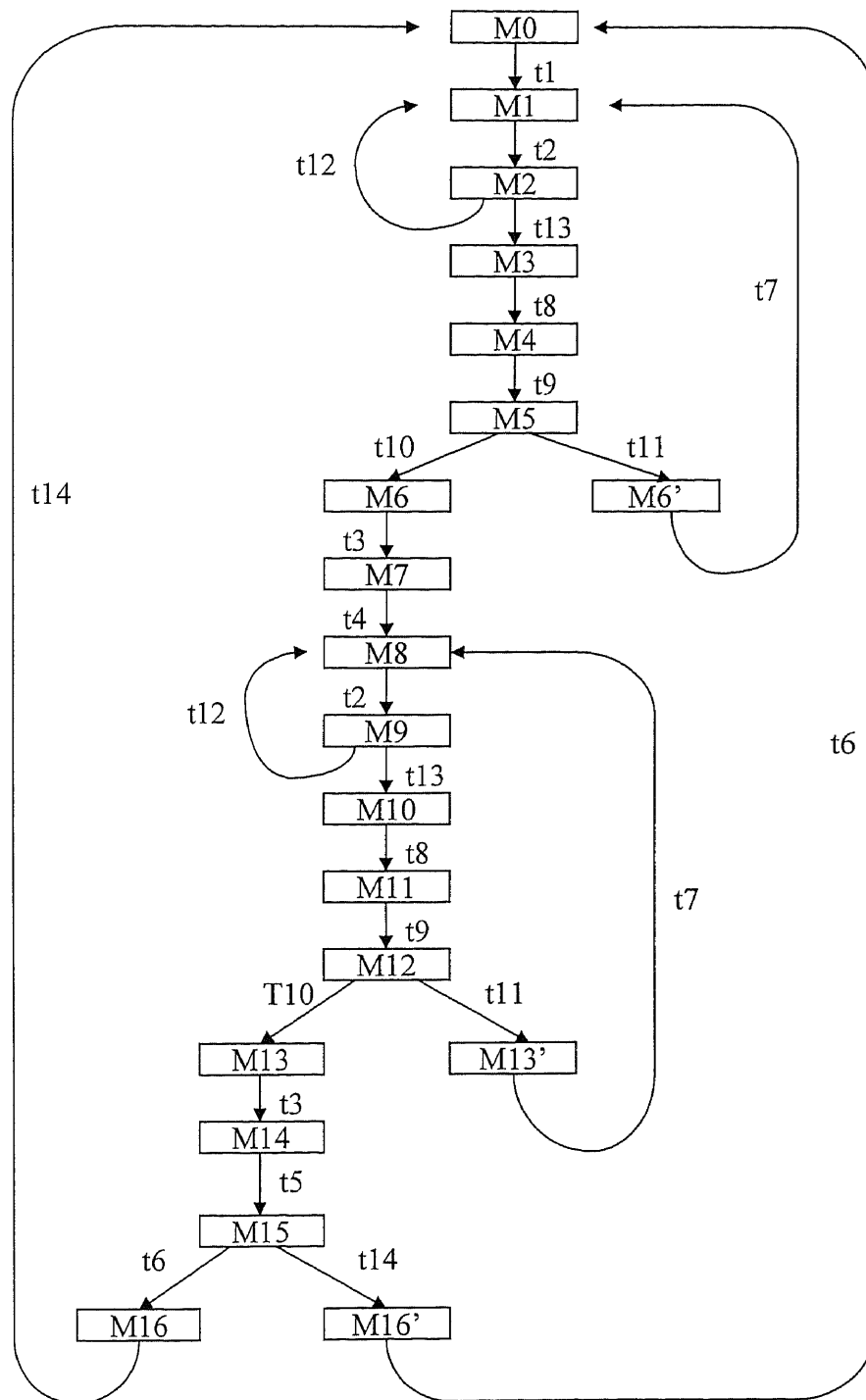


**Figure 4.5 a)** Transition Sequence for the CAN FieldBus Access Protocol, Part 1  
 (k=2)

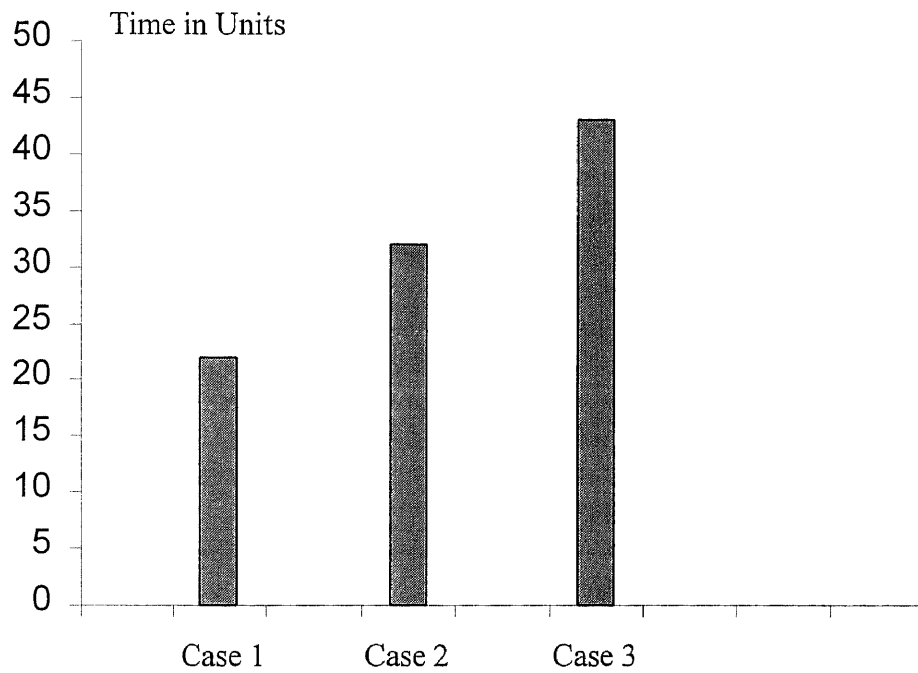




**Figure 4.5 b)** Transition Sequence for the CAN FieldBus Access Protocol, Part 2  
( $k=2$ )



**Figure 4.6** Reachability Graph for the CAN FieldBus Access Protocol ( $k=2$ )



Case 1: A successful communication. {Cycle time in Loop3(4) is 22 units}

Case 2: Loop1 is in effect once during a successful communication.  
 {Cycle time in Loop3(Loop4)+Loop1 is 32 units}

Case 3: Loop2 is in effect once during a successful communication.  
 {Cycle time in Loop3(Loop4)+Loop2 is 43 units}

**Figure 4.7** Effects of Some Factors on a Successful Communication  
 (k=1, Speed=2.5Mb/s, and 64bits/frame)

**Table 4.1** Source and Destination Place Description

<b>Place</b>	<b>Description</b>
p <sub>1</sub>	Idle, or initialization request
p <sub>2</sub>	Ready to send frame
p <sub>3</sub>	Frame sent and waiting for response
p <sub>4</sub>	Checked for more frame
p <sub>5</sub>	Ready to disconnect
p <sub>6</sub>	Ready to receive
p <sub>7</sub>	Frame received
p <sub>8</sub>	Frame processed
p <sub>9</sub>	Positive response sent
p <sub>10</sub>	Negative response sent
p <sub>11</sub>	Self-test
p <sub>12</sub>	Frame sent by sender
p <sub>13</sub>	The number of frames to be sent
p <sub>14</sub>	The number of the received frames
p <sub>15</sub>	Control

**Table 4.2** Source and Destination Transition Description and Time Delay

<b>Transition</b>	<b>Description</b>	<b>Time Delay</b>
t <sub>1</sub>	Establish channel connection	1
t <sub>2</sub>	Send frame	x*
t <sub>3</sub>	Response in OK	0
t <sub>4</sub>	More frame	0
t <sub>5</sub>	No more frame	0
t <sub>6</sub>	Disconnect	1
t <sub>7</sub>	Response in not OK	0
t <sub>8</sub>	Receive frame	0
t <sub>9</sub>	Process frame	1
t <sub>10</sub>	Send positive response and ready to receive	10**
t <sub>11</sub>	Send negative response and ready to receive	10**
t <sub>12</sub>	Negative response (the result of self-test)	0
t <sub>13</sub>	Positive response (the result of self-test)	0
t <sub>14</sub>	Recover counter size	0

x\*: It is indicated in Table 4.3.

10\*\*: It is assumed that a shortest frame (64bits/frame) is always used for response frames.

**Table 4.3** The Relationship among FieldBus Speed, the Length of the Frame and Time Delay in  $t_2$

FieldBus Speed	Bits/Frame	Time Delay in Units ( $t_2$ )
H2=2.5Mb/S	64	10
	128	20
	256	40
	512	80
H2=1Mb/S	64	25
	128	50
	256	100
	512	200
H1=31.25Kb/S	64	800
	128	1600
	256	3200
	512	6400

In Table 4.3, the different time delays in  $t_2$  are obtained according to the corresponding FieldBus speeds and different lengths of the frames

**Table 4.4** Loops and Their System Processes, Time Required to Complete the Process, and Cycle Time (Speed=2.5Mb/s, 64bits/frame)

<b>Loop No.</b>	<b>The System Process</b>	<b>Time Required to Complete the Process</b>	<b>Cycle Time in Units</b>
Loop1	Self-test is failed	$t_2+t_{13}+ t_{12}$	10
Loop2	Negative response	$t_2+ t_{13}+ t_8+ t_9+ t_{11}+ t_7$	21
Loop3(4)	Completion for a communication	$t_1+ t_2+ t_{13}+ t_8+ t_9+t_{10}+ t_3+ t_5$ $+ t_6 (t_{14})+ t_{14} (t_6)$	22

## CHAPTER 5

### SIMULATION

#### 5.1 A Timed Petri Net Simulator

In the previous Chapter, we assumed only one frame in each message for simplicity. When there are many frames in each message, the previous method is impossible due to the complexity of the resultant reachability graphs. For these complex cases, we can use a timed Petri net simulator TimedPNT to calculate the time required for a successful communication between the source and the destination stations.

TimedPNT [11], developed by NJIT in 1994, is a highly interactive graphical tool for drawing and simulating the systems with stochastic and deterministic delays. With the help of a user friendly menu, we can draw a Petri net easily using TimedPNT. The TimedPNT contains two different types of transitions, immediate and timed. For immediate transitions, there is no associated delay. For timed transitions, the user must specify the delay type as deterministic delay or stochastic delay. The former type is specified in this model. The time delays can be specified for each transition and a user can also design "priority" of each transition in order to determine firing sequence in the situation of conflicts between transitions. The net can be simulated using "step" or "run" simulation modes. Before simulation begins, the user can specify more than one terminating condition for the net.

- *End Condition:* A process of simulation can be terminated by End Conditions. There are two types of terminating conditions:
  1. System reaches a state of deadlock.



2. One of the following user pre-defined End Conditions is reached:

clock time; b) step number; c) place marking.

- *Three Reports:* There are three types of report generated by TimedPNT:
  1. The structural properties of the current net can be verified by using the "verify" option. It is very useful for ensuring the graphical drawing which is correctly interpreted by the tool for simulations.
  2. The utilization reports can automatically generate the utilization information of transitions and places after a simulation has been completed. This file contains utilization parameters of all places and transitions.
  3. With RUN or STEP mode simulations, log automatically generates a file containing a complete set of marking information during simulation, or the log report can keep the track of marking and transition firing information. Other details of the simulation, such as conflict resolution is also contained in the log file. The simulations with large numbers of steps will set large log files.
- *Utilization:* In a Utilization Report, net utilization parameters (transition and place) are calculated and recorded:

$$\text{For immediate transitions: Busy \%} = \frac{\text{No. of times fired}}{\text{No. of total steps}}$$

1. Transition Utilization: "number of times fired" and "Busy %"

$$\text{For timed transitions: Busy \%} = \frac{\text{total clock time holding tokens}}{\text{total run time}}$$

2. Place Utilization: "total tokens entered" and "duration occupied"

$$\text{For timed place: time occupied \%} = \frac{\text{total time occupied by tokens}}{\text{total run time}}$$

$$\text{For immediate place: time occupied \%} = 0 \%$$

## 5.2 Simulation of the Petri Net Model

TimedPNT is used to simulate the operational behavior of the CAN FieldBus access protocol so as to examine the utilization of the timed transition. In TimedPNT for this system, the cycle time for each successful communication is defined as:

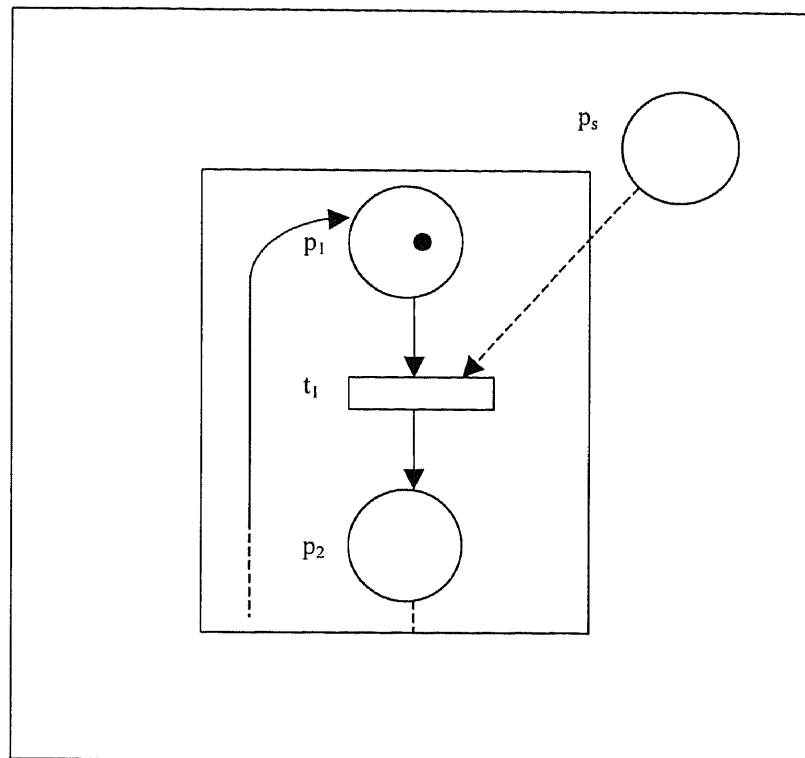
$$\text{Cycle time} = \frac{\text{Total system clock}}{\text{No. of total messages transmitted successfully}}$$

To understand the operation behavior of the system, we can get every step of the simulation from the log file. The report file indicates the percentage utilization of timed transitions and percentage utilization of all the places.

For the system model, we have associated the deterministic (constant firing time) delays with each transition. When the net is simulated in three different operational parameters, the corresponding results of the simulation are computed by TimedPNT.

## 5.3 Simulation Results

The Petri net in Fig. 4.1 is simulated with initial marking  $(p_1 \ p_6 \ kp_{13} \ p_{15} \ xp_s)$ . "k" is the number of the counter ( $k=1,2,3\dots$ positive integers). Note that  $kp_{13}$  means that "k" tokens are in  $p_{13}$ . "x" is the number of messages simulated. In order to simulate certain quantity of messages continuously, a simulating place  $p_s$  contains the number of messages to be sent as shown in Fig. 5.1. After the end of simulation, the final marking should be  $(p_1 \ p_6 \ kp_{13} \ p_{15})$  and  $p_s$  has no token (See Fig. 5.1).



**Figure 5.1** The Petri Net of the System for Simulation

It can be seen from Fig. 4.1 that transition  $t_2$  and place  $p_{13}$  play an important role in the overall performance of the net. The transition  $t_2$  is fired when a frame is sent to FieldBus and at any case the number of tokens in  $p_{13}$  is the number of the sent frames in one message in communication. Most of the transitions involved in communication process are located in the source and the destination stations. Some of the transitions are used for test, control purposes, and counter. The firing delay associated with transition  $t_2$  mainly depends on the underlying FieldBus parameters. FieldBus speed, and data size (or the length of a frame) to be transferred are two deciding factors for the delays of transition  $t_2$ .

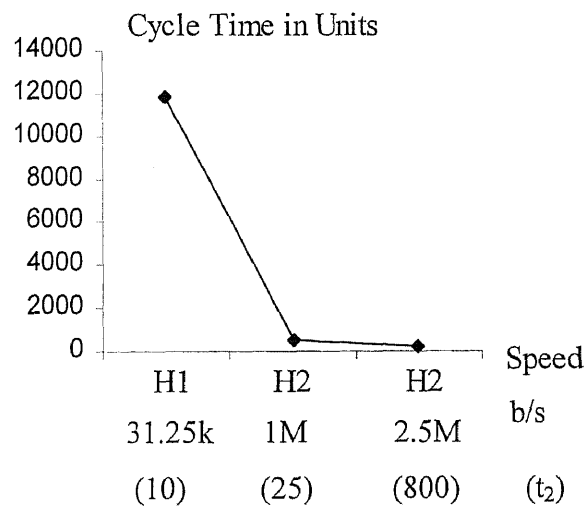
Since FieldBus speeds have H1 (31.25kb/s), H2 (1Mb/s) and H2 (2.5Mb/s), the delays with  $t_2$  could range from microseconds ( $10^{-6}$ ) to milliseconds ( $10^{-3}$ ). Tables 5.1-5.3 and Figs. 5.2-5.4 show the observations for these experiments. We have performed the following three simulation experiments:

- *Experiment 1:* Table 5.1 indicates that slower speed corresponds to a higher cycle time, and faster FieldBus corresponds to lower cycle time. Therefore, there is an inverse relationship between the speed and the cycle time.
- *Experiment 2:* Table 5.2 shows that when the same length of a frame (128bits/frame) is transmitted, different "k" values lead to the different cycle times at various speeds. As indicated in Table 5.2, with the same FieldBus speed and the same length of a frame, the corresponding cycle time is approximately doubled when "k" increases one fold. The relationship between these two cases is shown in Fig. 5.3.
- *Experiment 3:* In addition, when "k" is 4 and FieldBus speed is fixed at H2=2.5Mb/s, the different lengths of frames (i.e., 64, 128, 256 ...bits/frame) result in different cycle times.

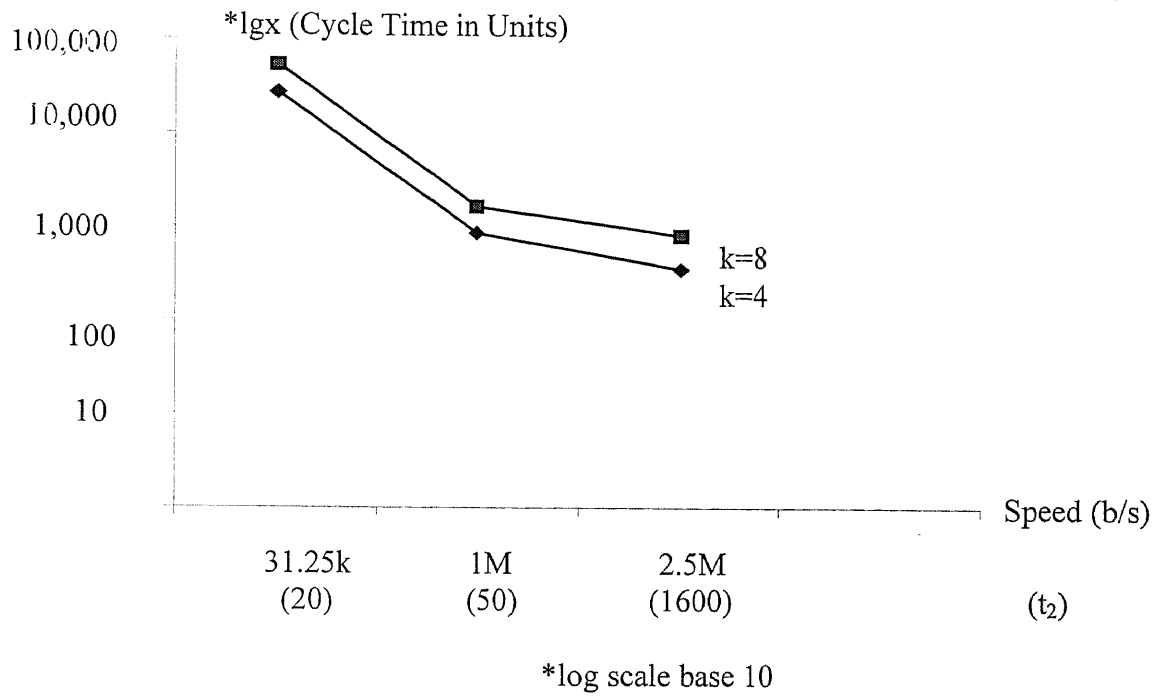
As shown in Figure 5.4, a respectively different cycle time can be obtained when a fixed counter and a certain speed are utilized in the system and the lengths of the frames are doubled. The cycle time increases relatively faster than the length of the frame as it is clearly observed from the Fig. 5.4 and Table 5.3. In these exhibits the cycle time is approximately doubled when the length of the frame is doubled. It demonstrates a non-linear relationship between the length of the frame and the cycle time.

**Table 5.1** Effect of the FieldBus Speeds (or  $t_2$  Delays) on Cycle Time  
( $k=4$ , and 64bits/frame)

FieldBus Speed	Time Delay in Units ( $t_2$ )	Cycle Time in Units
H2=2.5Mb/s	10	256
H2=1Mb/s	25	553
H1=31.25kb/s	800	11,843



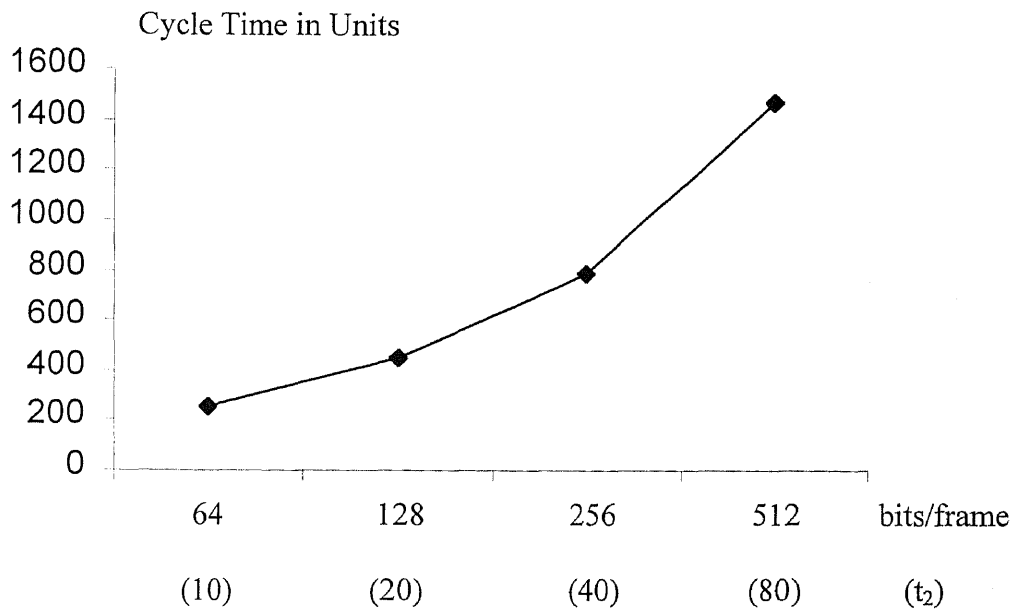
**Figure 5.2** Effect of  $t_2$  Delays on Cycle Time  
( $k=4$ , 64bits/frame)



**Figure 5.3** Effect of the Values of the Counter and  $t_2$  Delays on Cycle Time (lgx) (128bits/frame)

**Table 5.2** Effect of the Values of the Counter and the FieldBus Speeds (or  $t_2$  Delays) on Cycle Time (128bits/frame)

FieldBus Speed (b/s)	Time Delay in Units( $t_2$ )	Cycle Time in Units	
		k=4	k=8
H2=2.5M	20	352 (lg352=2.5465)	809 (lg809=2.9079)
H2=1M	50	855 (lg855=2.932)	1,656 (lg1,656=3.2191)
H1=31.25k	1600	26,877 (lg26,877=4.4294)	52,938 (lg52,938=4.7238)



**Figure 5.4** Effect of  $t_2$  Delays on Cycle Time  
( $k=4$ , Speed=2.5Mb/s)

**Table 5.3** Effect of the Lengths of the Frame (or  $t_2$ ) on Cycle Time  
( $k=4$ , Speed=2.5Mb/s)

Bits/frame	Time Delay in Units ( $t_2$ )	Cycle Time in Units
64	10	256
128	20	452
256	40	786
512	80	1474

## CHAPTER 6

### CONCLUSIONS

#### 6.1 Summary

This thesis presented a Petri net methodology for modeling and analyzing the CAN FieldBus access protocol. It develops a Petri net model for the entire system that clearly captures the events, states, and synchronization, concurrency, and dependency among these states and events. The model gives a clear understanding of the various events and operations involved in the CAN protocol. Using the reachability graph method, we have performed the behavioral analysis of the Petri net model. The obtained Petri net models include timed transition to model the time duration corresponding to the FieldBus speed and the length of each frame, and places to model the number of frames in a message. By this way, the important parameters of the CAN network — FieldBus speed, data size (the length of each frame) and the number of frames in a message that influence the performance of the CAN FieldBus are investigated through computer simulation. For the purpose of the simulation, a Petri net simulation package called TimedPNT is used.

The following observations were made based on TimedPNT simulation:

1. It takes a minimum of 22 time units to complete the communication process for one frame message between the source and destination (at FieldBus speed=2.5Mb/s,  $k=1$ , and 64bits/frame). It takes 10 or 21 more units time as the loop<sub>1</sub> or loop<sub>2</sub> is in effect once during processing a successful communication. Note that in a system process, loop<sub>1</sub> means that self-test fails and loop<sub>2</sub> means that the frame to be sent fails (or negative response is sent by the destination). We have found that these operations



contribute significantly towards the degraded performance of the CAN access processes. The overall process time can increase by 45% or 95% due to these events.

2. When a fixed counter and a certain speed are utilized in the system and the lengths of the frames are doubled, the cycle time increases relatively faster than the length of the frame and is approximately doubled. It demonstrates a non-linear relationship between the length of the frame and the cycle time.
3. During the transmission of the same length of a frame, the corresponding cycle time is approximately doubled when "k" is doubled.
4. A respectively different cycle time can be obtained when a fixed counter and a certain speed are used in the system. If the lengths of the frames are doubled, the cycle time is approximately doubled.

In summary, the method and models developed in this thesis are successfully utilized to performance analysis of the CAN FieldBus access protocol.

## 6.2 Future Work

- *Application of Colored Petri Nets*: In this thesis timed Petri Nets are used to model the CAN FieldBus protocol. Hence, the size of the Petri Net model increases very much when there are several types of the FieldBus protocols need to be modeled. As the size of the Petri Net model increases it poses difficulties for analyzing and simulating it to evaluate the performance. In the Petri Net literature it is very common to use a type of PNs called "Colored PNs". They can be used to offer a compact net model. Hence, further research is needed to apply and use Colored PNs by taking example of different types of FieldBuses and protocols. In Colored PNs places,

transitions, tokens can be given attributes or colors to model various properties. By applying Colored PNs for modeling the protocol, a compact PN model can be obtained which can be used to model various types of FieldBuses and protocols by simply changing the corresponding attributes of places, transitions, and tokens. For example, by associating attributes to tokens in appropriate places of the PN model presented in this thesis, the parameter "number of frames" in a message can be easily modeled and its impact on the performance of the protocol can be investigated. Similarly, various parameters in implementing a CAN FieldBus protocol can be modeled using a common PN structure but with different values for different colors/attributes of places, transitions, and tokens of the PN model.

- *Application of Real-Time Petri Nets:* Modeling and performance evaluation of a model is only one of the several stages of system implementation. After modeling and evaluating the performance of a FieldBus protocol using the PN model, it can also be used for the real-time control and monitoring of the FieldBus. In other words, PN models presented in this thesis can be extended to use them as a basis for developing the required discrete event controllers that monitor and control the FieldBus. The application of PNs for the control of flexible manufacturing systems and industrial automated systems is widely reported. Also, the application of PNs for modeling and performance evaluation of protocols is widely known. However, the application of the same PN model that is used for evaluating the performance of a communication protocol such as FieldBus protocol for developing real-time controllers is not reported yet. Hence, in the future research the PN models developed in this thesis can be

integrated with concepts such as Real-time PNs to not only control the FieldBus but also the industrial systems that use the FieldBus.

- *Development of Object-Oriented Simulators Based on the PN Models:* Object-oriented technology is often used to develop reusable software systems. Using object technology, the software is modeled as a set of communicating objects. An object typically contains both data (modeled as attributes of the object) and services (modeled as methods/operations of the object) and is a fundamental building block for programming a system using an object-oriented paradigm. PNs also model data in terms of places and services in terms of transitions. Hence, there is a correlation between a PN and an object. Due to this fact, the PN models presented in this thesis can be used to model a set of software objects that can be used for developing simulation and control software related to communication protocols in general and FieldBus protocols in particular. It should be noted that the combination of using the concept of Colored PNs and object-oriented technology may significantly help to manage the complexity associated with the application of PN models for a large scale industrial systems.

## REFERENCE

1. Foundation FieldBus, *Technical Overview*, Foundation FieldBus, Austin, Texas, 1996.
2. M. Santori, and K. Zech, "FieldBus Brings Protocol to Process Control," *IEEE Spectrum*, pp. 60-64, March 1996.
3. G. Paula, "Building a Better FieldBus," *Mechanical Engineering*, pp. 90-92, June 1997.
4. J. R. Pimentel, *Communication Networks for Manufacturing*, Prentice Hall, Inc, Englewood Cliffs, NJ, 1990.
5. G. Cena and A. Valenzano, "An Improved CAN FieldBus for Industrial Applications," *IEEE Transactions on Industrial Electronics*, Vol. 44, No. 4, pp. 553-564, 1997.
6. R. Zurawski and M. C. Zhou, "Petri Nets and Industrial Applications: A Tutorial," *IEEE Transactions on Industrial Electronics*, Vol. 41, No. 6, pp.567-580, 1994.
7. M. C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*, Kluwer Academic Publishers, Boston, MA, 1993.
8. M. C. Zhou and A. D. Robbi, "Applications of Petri Net Methodology to Manufacturing Systems," *Computer Control of Flexible Manufacturing Systems, Research and Development*, Edited by S. B. Joshi and J. S. Smith, pp. 208-230, 1994, USA
9. A. B. Mnaouer, Y. Fujii, and T. Sekiguchi, "Evaluation of the Normalized Proportional Allocation Scheme's Application to the Asynchronous Bandwidth in the FieldBus Protocol," *IEEE 6<sup>th</sup> International Conference on Emerging Technologies and Factory Automation Proceedings*, pp. 127-132, Sept. 1997.
10. K. G. Shin and C. C. Chou, "Design an Evaluation of Real-Time Communication for FieldBus-Based Manufacturing Systems," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 3, pp. 357-366, June 1996.
11. B. Liu, "A Graphical Tool for Timed Petri Nets Using Object Oriented Programming," Master's Thesis, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Jan. 1994.

12. K. Tindell, A. Burns, and A. Wellings, "Calculating Controller Area Network (CAN) Message Response Times," *Control Eng. Practice*, Vol. 3, No. 8, pp. 1163-1169, 1995.
13. S. Cavalieri, A. D. Stefano, and O. Mirabella, "Centralized versus Distributed Protocols for FieldBus Applications," *1995 IEEE 21<sup>st</sup> International Conference on Industrial Electronics, Control, and Instrumentation*, Vol. 2, pp. 1580-1585, Nov, 1995.
14. S. Cavalieri and O. Mirabella, "Enhancing Performance in FieldBus Communication Systems," *1996 IEEE Conference on Emerging Technologies and Factory Automation*, Vol. 2, pp. 609-615, Nov. 1996.