Modeling and Simulation of Industrial Networks With Timed Petri Nets

Gheorghe Sebestyen
Technical University of Cluj
Cluj. str. G. Baritiu nr. 26-28
Romania
gheorghe.sebestyen@cs.utcluj.ro

Péter Keresztes
Szechenyi Istvan University
Győr, Egyetem tér 1
Hungary
keresztes@sze.hu

ABSTRACT
The article presents a new modeling and simulation methodology, specially developed for the requirements of industrial communication networks. The methodology is based on an extended Timed Petri Nets model. The program developed on this methodology offers the possibility to test the real-time and reliability characteristics of industrial communication protocols.

Keywords: industrial networks, Timed Petri Nets, simulation, modeling

I. INTRODUCTION

Industrial networks represent a special category of digital networks adapted to the specific communication needs of a control system [5][6]. The communication protocols developed for these networks must assure an improved real-time response and a higher reliability [9]. Modeling and simulation tools used for general-purpose computer networks don't offer enough support to describe and to test real-time behavior and fault tolerance. This paper proposes a new modeling and simulation method based on Petri Nets, adapted to the special needs of industrial network protocols.

The application developed on the base of this method allows graphical description of protocols' behavior and network simulation under different load conditions. To demonstrate the advantages and facilities offered by this method, a number of case studies are presented.

II. INDUSTRIAL COMMUNICATION NETWORKS

Complex computer-based control systems need a communication infrastructure that connects field automation devices (e.g. sensors, transducers, actuators) with control and supervisory devices (e.g. PLCs, regulators and process computers). Industrial communication networks were specially developed for such purposes. These networks are optimized for the specific data flow of a control system [6], their behaviour is deterministic and real-time message delivery requirements can be guaranteed.

In a control system the correctness of a program execution is decided not only on the correctness of the calculus, but also on the fulfillment of time requirements. The communication, as part of the control chain, must fulfill the same requirements. The communication protocol must include some mechanisms, through which the delivery time of messages can be controlled. Usually messages are scheduled for transmission, off-line or on-line, based on their time characteristics: deadline, repetition period, transmission time, etc. With a protocol modeling and simulation tool, a designer can verify if the time restrictions for message delivery are accomplished, before the physical implementation o the system.

In many cases, control systems must work 24 hours a day, and any interrupt caused by defective components or communication failures can cause important losses. For this reason the industrial protocols must allow in-line disconnection of defective components and connection of new ones. The error detection and correction mechanisms implemented in the protocol must make communication errors transparent for the higher-level control application. Through simulation the designer can study the behavior of the network in the presence of such (simulated) errors. An important aspect in this case is the time delay caused by the presence of an error. Some protocols have time and space redundancies (e.g. message duplication, two communication lines, CRC- Cyclic Redundancy Code), through which communication errors can be masked.

III. PROTOCOL MODELING WITH PETRI NETS (PN)

Communication protocols involve complex mechanisms for synchronization, flow-control, routing, error detection and correction, etc. Abstract description, modeling and simulation are ways of controlling this inherent complexity [7].

Petri Nets represent a strong abstract formalism used to model discreet event systems. Petri Nets models are recommended to describe interaction, concurrency, parallel execution or mutual exclusion. These characteristics are typical for network protocols. But the classical formalism of Petri Nets cannot express properties connected with time (e.g. execution time, delays, synchronization, etc.). A number of extensions of the basic formalism were proposed, to add time behavior

for these models. There are three main approaches, called Timed Petri Nets (TPN).

The model proposed by Ramamoorthy [1] uses a function that attaches a firing time to transitions. A transition is fired in two phases: in the first phase tokens are consumed from the input places (as in the classical PN); the second phase starts when the firing time elapsed; in this phase tokens are generated in the output places. The model can describe atomic actions that are executed in a predefined time.

In the model proposed by Merlin [1] transitions are fired instantaneously (as in the classical PN), but in a static firing time interval. The moment when an enabled transition is executed is elected from the time interval, based on a probabilistic function. This formalism is adequate for sporadic and random event simulation.

The third model proposed by Sifakis [1] attaches a delay time to places. A token that arrives to a place will be delayed with the amount of time attached to it. The token can enable a transition only when the delay elapsed. Another version of this extension attaches delays to the arcs. In this case tokens have to wait different time intervals depending on the arcs they traveled. Tokens inside of a place can be identified if they are "colored". Colored Petri Nets represent a more general model that allows attachment of structured data to tokens.

Trying to model industrial communication protocols with one of these Timed Petri Nets extensions seamed difficult and in many ways incomplete. For instance the evolution of a TPN model depends only on some discrete events which are "consumed" during the firing of a transition. But in a communication protocol some actions are started not only on the occurrence of some events, but also based on the state of an entity (e.g. the state of the network), which is not modified after the action is executed. Another issue is that industrial protocols need different time semantics to express delays, transmission time, deadline, repetition period, etc.

To overcome these problems, a new TPN extension was developed. The following new elements were added to the basic formalism:

- transitions with execution time
- transition with probabilistic execution time
- transitions with enabling time
- re-triggerable transitions with execution time
- condition and non-condition arcs
- a global time counter

Transitions with execution time are used to model some actions that take place in a predefined period of time (e.g. transmission of a message or periodic generation of a new message). When the transition is fired, a counter is initialized with the execution time attached to the transition. The counter is decremented by every "tic" of the global time counter. The transition ends when its counter becomes zero. The tokens from the input places are consumed at the firing moment and the tokens for the output places are generated at the end of the transition. The transition is atomic, it cannot be interrupted or stopped by any event.

A transition with probabilistic execution time is similar with the previous one, but the execution time is randomly selected from a given interval. This kind of transition can be used to model sporadic events or actions with variable execution time.

A transition with enabling time is fired if the enabling conditions stay stable for a period of time specified by the value attached to the transition. This transition is useful for time-out or watchdog conditions modeling. For instance an error recovery procedure is started if an acknowledge message does not arrive in time. The re-triggerable transition can be used for similar purposes. The difference is that the transition is started when the enabling conditions are valid and its counter is reinitialized whenever the enabling conditions become again valid. The transition ends if the time between the last enabling condition and the next one is greater then the execution time of the transition. For instance the following situation can be modeled with this transition: a network node can transmit a new message if the line is free for a given period of time. If the transition ends, then it will generate a permission token to transmit a message.

Condition and non-condition arcs are used to enable or disable transitions based on the state of the input places. If an input place is connected to a transition with a condition arc then the transition is enabled if the input place has one or more tokens. If the transition is fired the number of tokens in the input place is not changed (the "state" of the location remains the same). A non-condition arc works in a similar way, but the transition is enabled if the input location has no tokens. These arcs are used to control the evolution of the TPN model based on the state of some entities rather than on some discrete events. For instance a lower priority message transmission (modeled by a transition) is conditioned by the absence of a higher priority message in the output buffer (modeled by a location). The transmission of the message does not change the state of the output buffer.

The global time counter is used to control the evolution in time of the TPN model. A time step is allowed when all the enabled transitions of the previous step are fired. When the time counter is incremented the local counters of the transitions in progress are decremented. A local counter is initialized with the time value attached to the transition when the transition is fired. If a local counter becomes zero, than its transition ends and tokens are generated into the output places. The global time counter can be used to mark with time-stamps the moments when a transition starts and ends. In this way concurrent events generated by the TPN model can be ordered in time. During the execution of a model, a transition may have different states. Its state depends on the

transition's type, on the enabling conditions and on its previous evolution. A transition with execution time may be in one of the following states: disabled, enabled, firing, in progress and transition end. A transition with enabling time has the following states: disabled, enabled and firing. Figure 1 shows the rules for state changes.

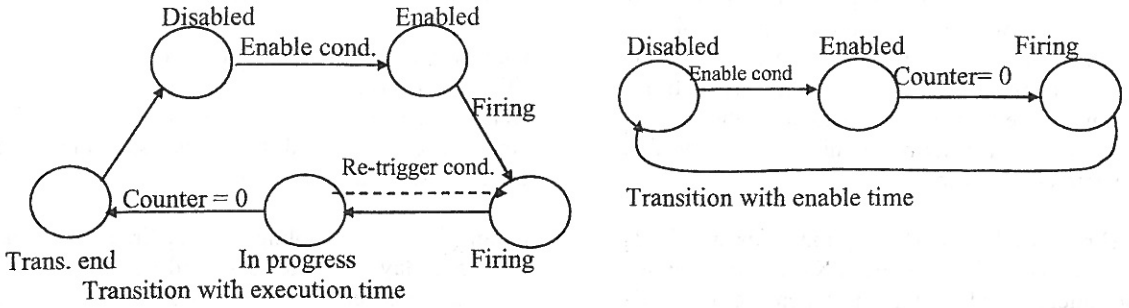

Figure 1 State changes for transitions

IV. TPN MODELING AND SIMULATION ENVIRONMENT

Based on the extended TPN formalism presented in the previous paragraph, a modeling and simulation environment was developed. This environment offers some useful tools for a communication protocol designer, such as:

- a graphical TPN model editor
- a library with some well-known protocol models and protocol building blocks
- a TPN model simulator

The simulator offers different possibilities to trace the execution of a TPN model: step-by-step, time-step, continuous (slow and fast), from a predefined starting marking to an ending one, etc. The same model can be tested under different conditions. Starting conditions are specified through the initial marking. Modifying the execution time of transitions may change time conditions.

The simulation window shows the steps made by the model during execution. The following information is displayed for every step: time-stamp (the value of the global time counter at the moment when the step was executed), marking (number of tokens in every location) and start or end of a transition. Through this information the designer can learn about the behavior of the model. Many design errors such as deadlocks, unreachable states, or deadline misses can be revealed. The maximum token number in a place can show the optimal dimension of a message buffer.

The library included in the environment contains some prototypes of typical industrial communication protocols, such as: token-bus, virtual token bus, CSMA/BA (Carrier Sense Multiple Access with Bit Arbitration), TDMA (Time Division Multiple Access), etc. These prototypes are a good starting point for more complex or detailed protocol models.

The primary goal of this environment was protocol modeling and simulation. But because of its generality in terms of formal descriptions, the environment can be used for a wider range of applications. It is recommended mainly for distributed control applications, because it offers modeling support for concurrency, event-triggered execution and real-time behavior.

V. CASE STUDIES OF INDUSTRIAL COMMUNICATION PROTOCOL MODELING AND SIMULATION

To show the advantages and limitations of the proposed TPN extension and of the developed environment, a number of representative industrial protocols were modeled and simulated. The analysis was focused on protocol characteristics that are specific for the category of industrial networks, such as: real-time behavior, determinism, reliability and fault-tolerance. These characteristics are determined mainly by the mechanisms used to control the multiple accesses on the network (MAC- Multiple Access Control protocol). For this reason the protocol models described with the TPN formalism express mainly the behavior of this protocol level.

There were selected four types of industrial protocols, with relevant access control mechanisms:

- Profibus – as a token-bus network,
- CAN – as a CSMA/BA network,
- P-Net – a virtual token-bus network, and
- TDMA – a reservation-based network

Figure 2 shows a part of the Profibus protocol model, responsible for the access control mechanism. Place L1 holds the messages waiting for transmission, L2 holds the right to transmit on the network (token) and L4 holds the token when the transmission right belongs to the other nodes from the network. Transition T1 models the transmission of a message, T2 models token passing, and T3, T4 simulate the delay of a token passing cycle through the network. In this simplified model the transmission right is transferred from the current node if there are no messages pending in the message buffer. In a more detailed model the time, during which a node holds the token, was limited.

Some other time restrictions were modeled in the case study, such as: maximum token passing cycle, maximum

time delay for an acknowledge message and maximum recovery time for a lost token. The protocol was tested also under simulated transmission errors.

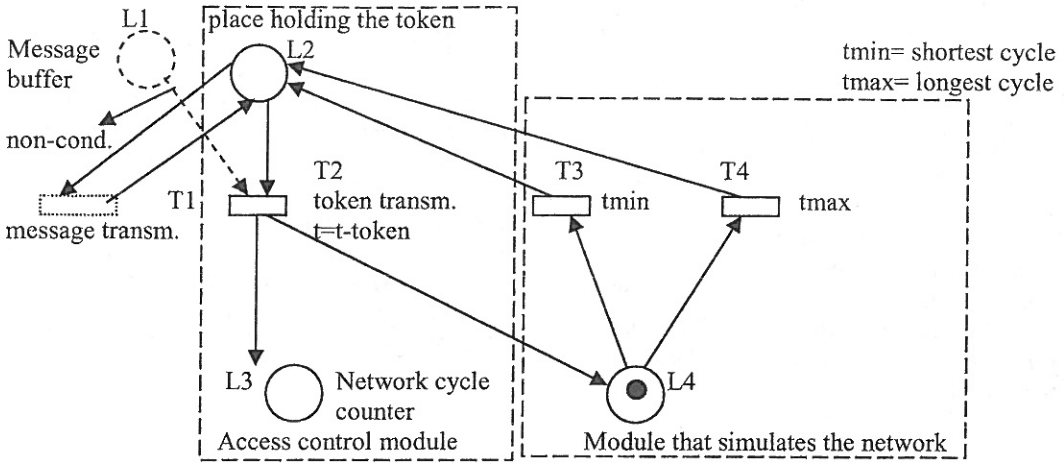

Figure 2 Access control mechanism

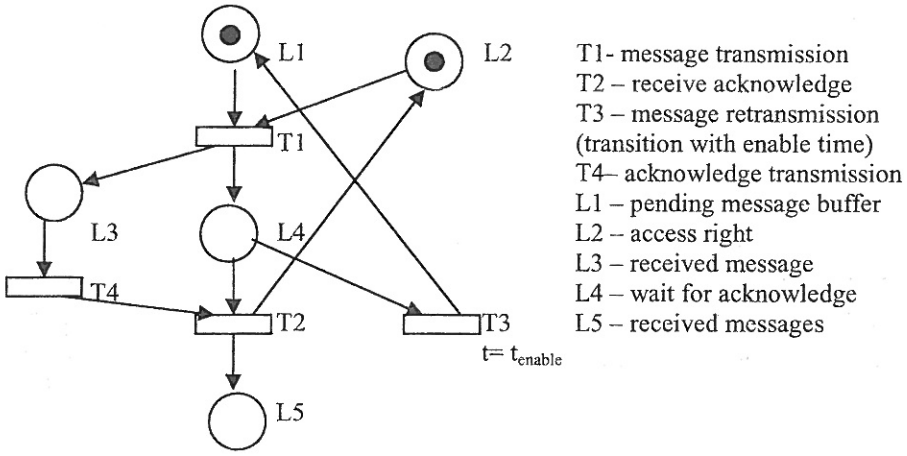

Figure 3 Limited acknowledge time

Figure 3 shows a recovery mechanism used to retransmit a message if its acknowledge message does not arrive in a predefined time. In the model, transition T3 initiates a message retransmission if the acknowledge generated by transition T4 is delayed.

The other industrial protocols were modeled and simulated in a similar way. A detailed presentation of these case studies is presented in a technical report [10]. The conclusions of these case studies are the following:

- the extended TPN formalism offers a good support for the analysis of industrial protocol characteristics
- basic protocol mechanisms can be tested in an early design phase
- the complexity of the TPN protocol model gives a good measure of the complexity of the protocol itself
- there are some limitations inherent to any classical Petri Net formalism, such as: limited model structuring possibilities and complex enabling conditions are difficult to express; these drawbacks can be partly overtaken with the use of colored PN.

VI. CONCLUSIONS

Industrial communication protocols have specific characteristics, which imply the use of adapted tools for modeling and simulation. This paper proposes an extended TPN formalism that offers good flexibility in expressing these characteristics. This formalism is also recommended for distributed control systems.

A modeling and simulation environment was implemented based on this new formal description method. The case studies made on some representative industrial protocols showed the advantages and also the limitations of the proposed formalism.

VII. ACKNOWLEDGEMENTS

The work was possible due to the support given by the Joint Hungarian – Romanian Intergovernmental S & T Co-operation Program under grant 13/2002 and contract No. 18051/2003 (code RO-9/2002 and HU-17/2002).

VIII. REFERENCES

[1] FBowden, "Modelling Time in Petri Nets" , *The 2-nd Australia-Japan Workshop on Stochastic Models*, Goald Coast, 1996

[2] C. Cardeira, F Simonot-Lion, M Bayard, " Intelligent Field Devices and Field Buses: Impact on Applications Design Methodology", *Studies in Informatics and Control Vol 4 No. 3, pp 255-262*, 1995

[3] R. Crowder, "Fieldbus Performance", *Technical Reprot, Ship Star Associate*s, 1996

[4] C. Eriksson, M Gustafsson, H Thane, "A Communication Protocol for Hard and Soft Real-Time Systems", *8th IEEE Euromicro, Real-Time Workshop*, L'Aquila, Italy , 1996

[5] L.B. Fredriksson, "Controller Area Networks and the Protocol CAN for Machine Control Systems", *Mechatronics , Vol. 4 No.2 pp 159-192*, 1992

[6] S. Koubias, G.Papadopoulos, "Modern Fieldbus Communication Architectures for Real-Time Industrial Applications", *Computers in Industry journal (ELSEVIER) Vol. 26 pp.* 243-252, 1995

[7] Pasadas R. , Almeida L. Fonseca J. 1997, " A Proposal to Improve Flexibility in Real-Time Fieldbus Networks " IFAC SICICA '97 , *3rd IFAC Symposium on Intelligent Components and Instruments for Control Application , Annesy, Franta, 1997*

[8] P. Raja, J Hernandez, L Ruiz. F Guidec, J.D Decotignie., "Simulating Fieldbus Applications With DRUGH Simulator", *Computers in Industry (ELSEVIER) No. 27, pp.43-51*, 1995

[9] G. Sebestyen, K. Pusztai, "New Networking Technologies in Control Applications", 2nd RoEduNet Conference, 2003