Computer Network Simulation Modeling Based on an Object Oriented Petri Net

http://dx.doi.org/10.3991/ijoe.v12i02.5039

CHEN Xinhua, SUN Ya-ni
Sichuan Information Technology College, China

***Abstract*—This paper first briefly introduces a Petri net, and then studies it in detail according to the selected object-oriented Petri net modeling method. This paper improves the object-oriented Petri net modeling method and studies the modeling and operation steps in detail. Finally, this paper builds a computer network TCP/IP protocol model based on an object-oriented Petri net. The model proves to be a good simulation of the computer network and the improved modeling method is also a valid method for further object-oriented Petri net modeling.**

***Index Terms*—Computer Network, Simulation Modeling, Petri net.**

I. INTRODUCTION

Petri net is a graphical representation of a combination model. It has advantages such as being available and easy to understand and has easy usability. Compared to other networks, Petri net has a unique advantage in the field of description and analysis. At the same time, Petri net is a strictly defined mathematics object [1-3]. With the aid of the mathematical development of Petri nets analysis methods and techniques, it can be used for static structural analysis as well as for dynamic analysis of the behavior. Petri net modeling technology can be used to simulate systems with concurrency, asynchrony, distributed, and uncertain parallelism features. Petri net has become the most promising modeling tool [4].

II. THE CATAGORY OF OBJECT-ORIENTED PETRI NET

Object-oriented Petri nets can be roughly divided into three categories.

Treat token as an object, which has identification, properties and methods. When the change is ignited, the corresponding method on the token will be called [5].

The internal behavior of the object can be represented by Petri net, such as G-CPN (G-colored Petri net). In this method, the Petri net identifies the current state of the object. Methods can connect to changes or libraries that make up the system communication. When the relationship between objects is defined statically, the global model of the system can be defined.

The third method combines with the characteristics of the first two methods. In this case, the token of the system net was treated as an object. The object can be a Petri net that is used to describe the behavior of the token. Additionally, the token in the subnet can also be a Petri net and recursive, such as OPN (Object Petri Nets).

According to the different needs, the object-oriented method combined with Petri nets vary in forms, such as OPNets, COOPN / 2, G - Net.

III. OOPN MODEL

The OOPN model belongs to the whole system. In object-oriented Petri nets, the relationship between objects is very vivid. The whole system is composed of multiple layers and the message translates between different layered controller objects.

A. The external structure of the object in the system

An objects' external structure is mainly used to receive messages from the controller or be sent to the controller interface. In the interface of In and Out, if different objects are sending the same message to an object at the same time, the object of the message is received in the interface automatically from an a priority message queue; the object accordingly reaches a priority order of execution. Similarly, a message being sent in the interface is likely to produce a message queue (this is not a priority message queue, priority of the message object or other additional information is described by the following) given by the control structure.

B. The Internal Structure of the Object in the System

The internal structure of the object is its own attributes and its method of processing the data, shown in Figure 1. An object's properties are determined by the place. The properties can be a specific data type (such as integer, character, etc.) or an abstract data type (such as a data structure, etc.). Methods are represented by the transition. When represented in a computer, only get and sent in an object-oriented Petri net system are the parent class of all the objects. In this class, the properties of tokens and transitions get through the sent and get methods. Other objects inherit this class to implement their own function. For example, message objects inherit class, and receive message objects instantiate the message class after reconstruction to get the message in the message receiving object implemented in time.

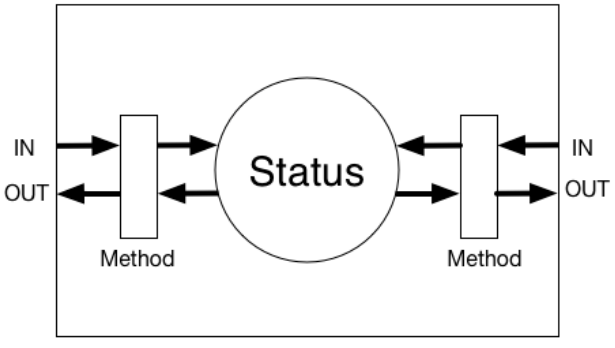

Figure 1. The Structure of the Object

C. Control Structure of Objects

By adding a control structure in each level of the system, the control structure is responsible for all objects or the lower layer control structure of forwarding messages. If the received message belongs to this layer, then this layer's control structure is responsible for the processing of the message (similar to the message processing in TCP/IP protocol). Furthermore, the control structure will add some additional information to the message, such as forwarding destination (i.e., to make a request to which object), priority coefficient and message sender (in this model, message transmission between objects is also a kind of object), or else the message will automatically forward to the next layer. All additional information about all operations is done by the controller structure of the message sender layer. Each other control structure in the process of message transfer only verifies or forwards the message, shown in Figure 2.

With such processing, it reflects object-oriented low coupling between modules to make each object's responsibilities clearer. The modify business process does not need to modify the program source code. For example, if you need to add a business process, you only need to add a class for class place, unless each object method of processing the data needs to be changed when the class place must be modified. But then you only need to modify the class without making changes to the business logic code. An object-oriented Petri net model, therefore, to a great extent will improve the scalability of the system. This is a general description of the system through a hierarchical description method, narrowing the scale of the system modeling and decomposing the complexity of the model, as shown in Figure 3. In this kind of practice with some object-oriented language such as a C++ programming process, the designers first focus on the structure of the local individual classes and objects and then look at the whole design of the application process.

IV. OBJECT-ORIENTED PETRI NET MODELING PROCEDURES AND STEPS

A. Object-Oriented Petri Net modeling procedure

The overall modeling procedure is shown in Figure 4.

First of all, we must make clear the user requirements and the main function of the system.

Because we want to use object-oriented Petri net, which is a modular layered approach to system modeling, it is necessary to decompose the requirements of the users and divide the system in logic.

In order to use the grammar of the object-oriented Petri net for modeling, we must break down the object-oriented Petri net for its syntax, presentation and build.

A Petri net method has the most prominent advantages for establishing the model for syntax analysis, and it is helpful for minimizing the loss caused by errors. So the next step is model analysis by analyzing the validity of the model outputs and integrity and other features.

Design system in detail or turn to (3) to further optimize the model.

B. Object-Oriented Petri Net Modeling Steps

The process is roughly divided into the following six steps:

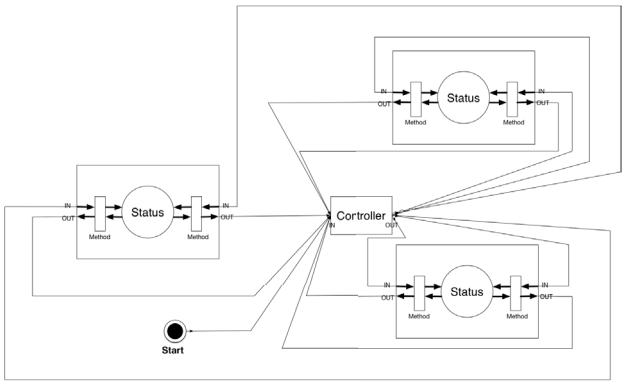

Figure 2. The relationship between Objects and Controller

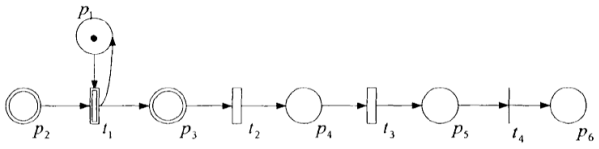

Figure 3. The Structure of the Controller

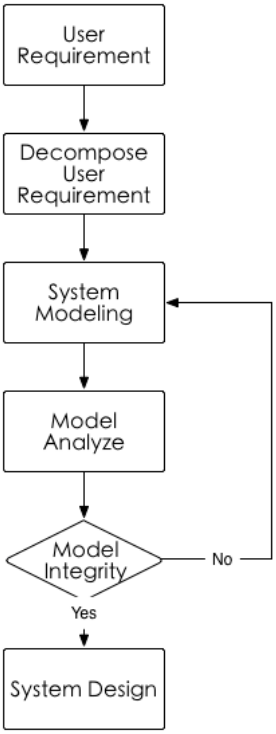

Figure 4. System Modeling Process

(1) Build the hierarchical model. Divide the research system into several subsystems and determine the relationship of each subsystem. Then describe the relationship with the roots, trunk, and base class as a tree structure hierarchical model.

(2) Establish the object model. According to all levels, list the names of the objects, characteristics, methods, and messages (the operation). In the discussion level, each subsystem is an object.

(3) Establish object relationship tables. According to the specific details of each subsystem, establish the event list within the system and through analysis of the event, establish the object relationship tables.

(4) Establish event tables. Through the object model analysis of the conditions and consequences of each event, create the event tables

(5) Determine the initial identification of the system. Determine the instance of each object. To determine the initial state of the instance, put each token into a place corresponding with its initial state.

C. OOPN System Operational Steps

When the system is running, first by an external issue order send a message to the controller with the initial identification; as this step is completed, the rest of the operations are automatically completed by each object. Generally go through the following steps:

(1) With the initial identification of the controller, analyze the received message is sent to the corresponding object and then transfers control to the current object.

(2) As each object receives the message, put the message in the corresponding input message queue interface and form an input message queue.

(3) The objects get messages from the message queue, according to the priority order, and change the state of the object.

(4) When step (3) is completed, put the corresponding message in the message output interface, form an output message queue list, and send it to the controller of the current level.

(5) The controller first adds additional information (such as priority, level, time property, etc.) to the received message, and at the same time decides whether the destination of the sending is the current level. If it is, the controller will send it to the corresponding object or forward it to the next level controller. It needs to be emphasized that the additional information the controller adds to the message is restricted to the controller of the object's level. The inter-median controller only forwards the message, and the receiving controller analyzes the additional message (such as time, priority of the message object, message sending objects, etc.). Eventually the message will be sent to the destination (message receiving object). When implemented with a computer, useful information can be put into the object table in the database, and the system can analyze the long term data in the database on a system optimization basis.

(6) The system will repeat the steps described above, until the task is complete.

V. THE TCP/IP PROTOCOL OBJECT PETRI NET MODEL

TCP/IP protocol provides the network with a reliable, end-to-end byte stream communication transport layer protocol, which has become a universal standard for network communication. TCP/IP protocol relies on a message confirmation response mechanism and timeout retransmission mechanism to guarantee packet delivery to an application layer in order by a sliding window to control a message's sending and receiving traffic. Thus, the TCP/IP protocol has four sub models: receiving model, sending model, sliding window control module, and timeout resending model. Only the "APP layer entry", "APP layer export", "IP layer entry", and "IP layer export" are externally visible. Other ports are only used for internal sub models to communicate information and interact. The receiving and sending models work together to complete the message confirmation and response mechanism. The whole process, including connection request, answer, set up, and transmission, of the TCP/IP protocol can be seen in Figure 5 and Figure 6.

Base on the TCP/IP protocol OOPN receiving and sending model, the TCP/IP protocol OOPN model is shown in Figure 7 and Tables I and VI.

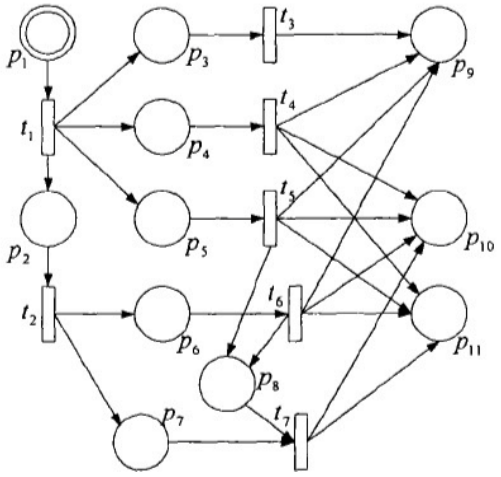

Figure 5. TCP/IP Protocol Receiving OOPN model

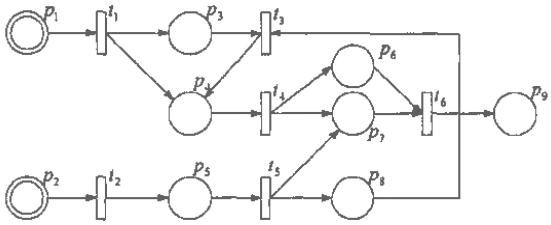

Figure 6. TCP/IP Protocol Sending OOPN Model

TABLE I.
DEFINITION OF PLACES IN TCP/IP PROTOCOL RECEIVING OOPN MODULE

Place	Definition	Place	Definition
p_1	Receiving port	p_7	Connection Nonexist Message
p_2	Message	p_8	Receiving Cache
p_3	Connection Request Message	p_9	APP Layer
p_4	Connection Response Message	p_{10}	Timer
p_5	Connection Release Message	p_{11}	Sending Cache
p_6	Connection Exist Message		

TABLE II.
DEFINITION OF TRANSITIONS IN TCP/IP PROTOCOL RECEIVING OOPN MODULE

Transition	Definition	Transition	Definition
t_1	Message identification	t_5	Release Connection
t_2	Distinguish Connection Message	t_6	Increase Cache
t_3	Upload to APP Layer	t_7	Decrease Cache
t_4	Process Connection Response Message		

TABLE III.
DEFINITION OF PLACES IN TCP/IP PROTOCOL SENDING OOPN MODULE

Place	Definition	Place	Definition
p_1	Sending Message Port	p_6	Sending Ready
p_2	Receiving Message Port	p_7	Sent Cache
p_3	Cache	p_8	Cache Available
p_4	Cache Available	p_9	Sending Message Port
p_5	Clear Message		

TABLE IV.
DEFINITION OF TRANSITIONS IN TCP/IP PROTOCOL SENDING OOPN MODULE

Transition	Definition	Transition	Definition
t_1	Check Sending Cache	t_4	Decrease Sending Cache
t_2	Receiving Message Classify	t_5	Increase Sending Cache
t_3	Cache	t_6	Sending Ready

TABLE V.
DEFINITION OF PLACES IN TCP/IP PROTOCOL OOPN MODEL

Place	Definition	Place	Definition
p_1	APP Layer Entry	p_8	IP3
p_2	IP Layer Entry	p_9	IP0
p_3	Cache	p_{10}	IP Layer Exit
p_4	Sending Cache	p_{11}	Timer
p_5	OP1	p_{12}	Redeay to resent
p_6	IP2	p_{13}	APP Layer Exit
p_7	IP		

TABLE VI.
DEFINITION OF TRANSITIONS IN TCP/IP PROTOCOL OOPN MODEL

Transition	Definition	Transition	Definition
t_1	Process APP server	t_6	Define TCP Package
t_2	Process IP message	t_7	Resent
t_3	Send Connection	t_8	Processing Module
t_4	Increase Sending Cache	t_9	Timer
t_5	Receive Connetion	t_{10}	Sent to APP Layer

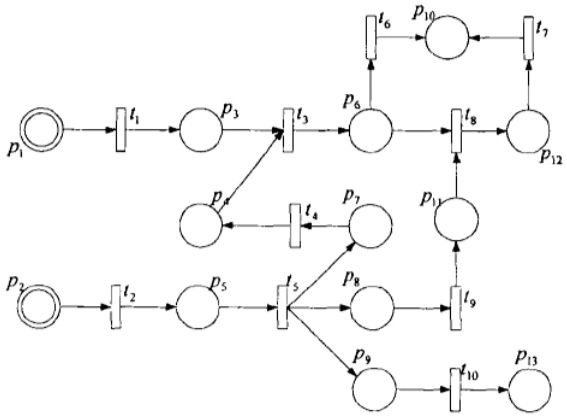

Figure 7. TCP/IP protocol OOPN model

VI. THE CONCLUSION

For a large, advanced and complex network control system, modeling and performance analysis is very difficult. Given that Petri net is a systematical graphical language with powerful function to describe and analyze complex systems, this paper used an object-oriented Petri net to model and analyze a computer network. With the improved modeling method proposed in this paper, a TCP/IP protocol OOPN model was built. It proved the validity of the modeling method and the effectiveness of the TCP/IP protocol OOPN model.

REFERENCES

[1] Bukowiec, A., & Doligalski, M. (2013). Petri net dynamic partial reconfiguration in fpga. Lecture Notes in Computer Science, 436-443. http://dx.doi.org/10.1007/978-3-642-53856-8_55

[2] Cheng Li, Zhang Jianxin, & Zhao Hailong (2013). The method of simulation based on stateflow. Computer simulation, 30 (2), DOI:10.3969/j.issn.1006-9348.2013.02.087. 378-382.

[3] Dmitry A. Zaitsev. (2013). A small universal petri net. Eprint Arxiv, 128, 190-202. http://dx.doi.org/10.4204/eptcs.128.22

[4] Fang Huan, Fang Xianwen, & Wang Lili (2014). Review of the research on the reliability analysis of Petri net. Computer science, 41 (7), 40-44.DOI:10.11896/j.issn.1002-137X.2014.07.007.

[5] Li Xiaozhong, & Zhang Delong (2013). The Petri system based on object oriented BPM net. Journal of Jiangsu University: Natural Science Edition, 34 (3), DOI:10.3969/j.issn.1671-7775.2013.03.010. 298-303.

AUTHORS

CHEN Xinhua is with the Sichuan Information Technology College, CO 628040, China.

SUN Ya-ni is with the Sichuan Information Technology College, CO 628040, China.

Submitted 17 September 2015. Published as resubmitted by the authors 23 January 2016.