LOKI ++: A PETRI NETWORKS SIMULATOR FOR PERFORMANCE EVALUATION

Paolo Giuffrida

University of Catania
Italy
yattapaul@katamail.com

Abstract

In this paper, Loki ++ a Petri Networks simulator is presented. The program, implemented in C + +, aims to create and analyze models which represent simple or complex systems relating to different fields of study.

In addition to graphic design of the network, the tool also allows the simulation of a system, and the monitoring of its evolution. The Petri Networks simulator is a useful tool for performance evaluation of a system. It allows us to know what the system can offer under certain operating conditions, in order to delimit the application area where it can be used successfully.

Finally the program is applied to study a model for Bluetooth characterized by 7 Slaves.

I. INTRODUCTION

Petri Networks, proposed in 1962 by Carl Adam Petri, are a powerful tool for the description and analysis of concurrence and synchronization in parallel systems. They represent a formalism widely used to model and describe the dynamic evolution of parallel, not deterministic processes.

In the description of a process (production, organizational, etc.) often we need to represent sub-processes or activities that may be performed simultaneously, that is, in parallel with each other, but not independently of each other: it could happen that a given passage or a certain phase of the process may not occur or can not be activated until other phases or activities are completed or not occur for certain conditions.

There are two main reasons that make Petri Networks useful: first they allow us to give the model of a system in a rigorous form in order to remove any ambiguity in the representation. This is important for the analysis and verification on the behavior of the system. Second, the Petri nets formalism is liable to a graphical representation which is rather spontaneous.

The constituent elements of a Petri Network are: places, transitions and arches. [1, 2]

The place can contain one, zero or more tokens, represented by a bullet inside the circle, which means that the semantic condition is valid. The totality of the marked places describes the status of the process.

The transition instead represents the "active" component; an event which changes the status of the network. A Petri Network evolves through the firing of its transitions. A transition "fires" when all of its input places are marked. When a transition is activated, tokens are removed from the places before the transition and they are placed in each of the output places of the transition.

A particular type of networks place/ transition (P/T) are timed Petri nets.

The ordinary Petri nets does not include any concept of time. With this class of networks, it is possible to describe the logical structure of a system, but not its temporal evolution. Various extensions to the networks are created to introduce the variable time.

In general, there are two distinct types of timed networks:

Deterministic Petri Networks where X is a variable defined which represents the time of completion of the activity. The transition which represents the activity after being empowered is such that the departure or the shooting will not be immediate but will depend on the time in x. [3]

Stochastic Petri Networks where X, defined on a generic transition, is not a deterministic, but it is in fact a random variable representing the feature "use" of the transition.

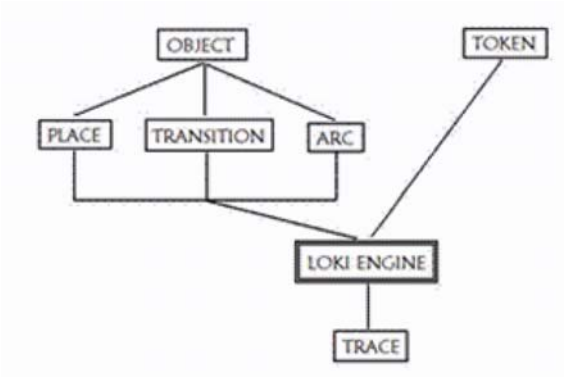
Adding the specific time, Petri Nets, PN, are used for the Performance; Petri nets simulator is a useful tool for the evaluation of processes and networks. [3]

The paper is organized as follows. In the next section we describe the architecture of the simulator. In Section III we will discuss the implementation of the software. In Section IV we state our graphical interface about Loki + +. In Section V we describe the application example of a Bluetooth network. Finally, in Section VI, we state our conclusions and give directions for future work.

II. LOKI ARCHITECTURE

An initial analysis about the problem showed the main needs in the development of the project Loki: the possibility to use Petri nets in all their aspects and the need to make the simulation speed. In particular, for the second requirement, it was noted as major delays occur in the functions dedicated to the display of the network and its animation; so a study on the method to optimize the most critical functions has led us to the conclusion (confirmed by result) that the faster method for viewing was to create a single class containing the various characteristics of representable objects and to insert all instances in a single list so they can recall in a single for cycle.

Is now shown the block diagram of the Loki simulator.



There are at top two classes Object and Token. The Object class contains all the elements of the network: place, transition and arc that go along with a token together in the "engine" of the simulator. At the end of the simulation the program creates a file showing the "history" of each token: you can see the places and transitions cross, and the time used to run.

III. SOFTWARE IMPLEMENTATION

Let's go to analyze in detail the features of the simulator and the criteria used for the deployment.

The code was written in C ++ using the functions offered only by Windows API and the STL, a standard library which exports templates of items such as lists, queues, stacks.

In DrawObject.cpp class, as mentioned above, there are all the different characteristics of representable objects: places, arcs and transitions.

All instances of this class are then inserted into a single list:

```
ObjList OL
```

Each DrawObject object contained in the list OL, exports a method Draw. With this method you can, knowing the type (Type is a member of the class), draw the object:

```
(DrawObject.cpp)
void Draw(HDC hDC)
{
      switch(Type)
      {
            case ID_PLACE:
                  ...
            case ID_TRANS:
                  ...
            case ID_ARC:
                  ...
      }
}
```

To make the simulation we have preferred to use a thread so that the same simulation was performed independently by button interface:

```
(CLoki.hpp)
      case IDC_BUTTON_RUN:
      ...

 hSimulThread=CreateThread(0,0,g_Run,NULL,0,N
ULL);
```

IDC_BUTTON_RUN is a message sent from the press of the key RUN; g_Run function only serves as an interface between the thread and function Run in the Loki class. This does nothing more than calling the Step function cyclically and checking that the simulation is not finished.

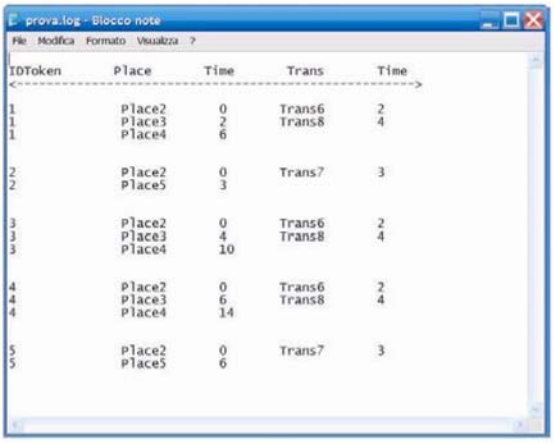
Step function is the heart of simulation; to make clear development is convenient to use a C-like language:

```
Step()
{
   We monitor the timers transitions ();
   We monitor and resolve conflicts ();
   for(each arc active)
   {
      We subtract a token to a place from
which it parts ();
      e activate the animation of the arc ();
      }
   We do the animation for each arc ();

   for(each active transition)
   {
      We enable the arcs connected ();
      We activated the animation of the arc
();
   }
   We do the animation for each arc ();
   for(each arc active)
   {
      We add a token to the place where it
arrives ();
   }
}
```

Within the function Step cycles, in this new version of Loki, the various information needed to know the "history" of each token are stored. For each token on our Petri network the program creates a new object from class token.cpp. Each shift from a place or a transition is stored in the variables of the class Token: int IDPlaces [100] and IDTrans int [100], so that at the end of the simulation we know the entire trace of tokens with their time.

In creating the file report, we have chosen the best format so you can easily import and modify the file through Excel.

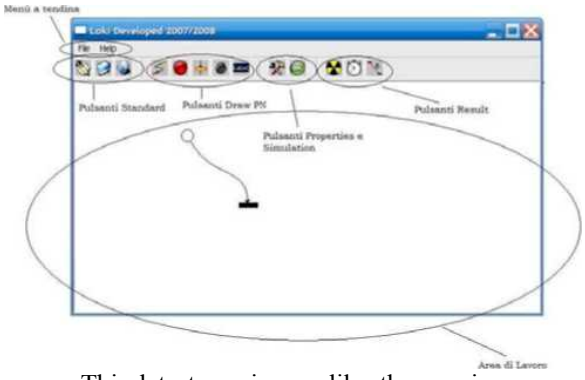


IV. GRAPHICAL INTERFACE

Let's go now to analyze the graphical interface of our simulator. The main window is shown in the next picture.

We note first that the graphical interface presents only two drop-down and, under these, a great toolbar.

Indeed, the program has been designed especially for use with mouse and through the use of buttons. Under the buttons, we find the work area to draw our Petri network and where the various simulations and analyses will be performed.



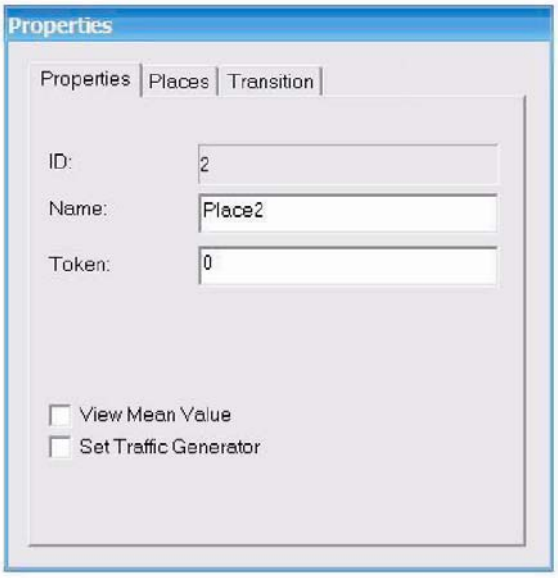
This latest version, unlike the previous ones, was developed with the goal of implementing new types of advanced features which allow the possibility to obtain more detailed information regarding the results of a simulation on a generic Petri network model studied by the user.

We have to pay attention, at this point, some properties of the simulator.

When we draw into the work area of the simulator the Petri Network to be examined you can set certain properties of places, arches and transitions.

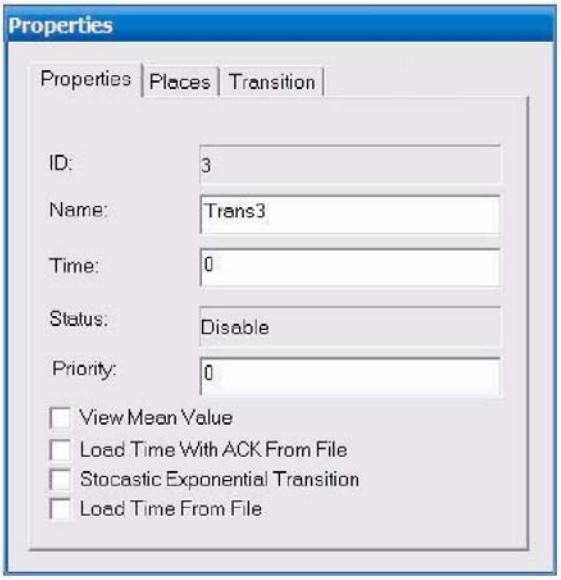
The folder properties, which is activated by the same name icon allows you to change some settings of the elements already drawn in according to the properties of Petri nets extended or stochastic generalized.

Once we activate the window, we can notice that it is divided into three parts or dialog windows: Properties, Places and Transitions.



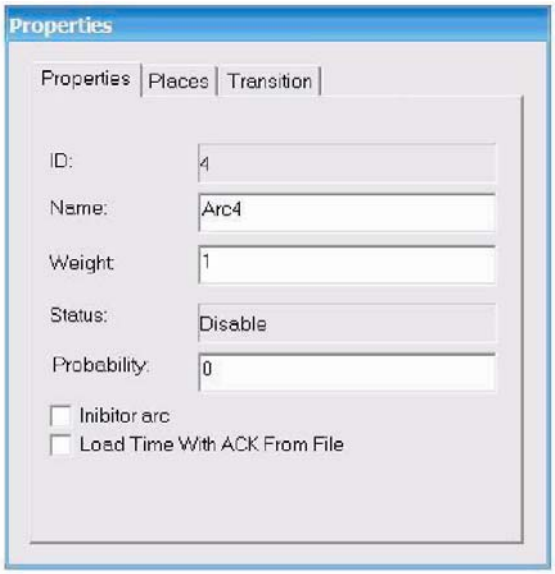
The first dialog window shows the properties for any object within the workplace, after the selection, making distinctions between the various objects and their characteristics. If you select, for example, a place you can see its features and possibly you can change them. Selecting a place the features will be ID, Name (which are present for all), and Token. ID is not changeable and chosen by the program, while Name, which can be by default, need to give a mnemonic identifier to the object. Token needs to indicate the quantity of token in that place. In addition there is the possibility to select the View Mean Time option in order to calculate the average number of steps per second in place, this option is present also with the transitions.

It includes a further option, Set Traffic Generator that allows you to create a traffic generator for the token.



In the case of selection of a transition as well as ID and Name options will be present Time, and Priority Status. Time is the time in seconds of timeout that we should wait before shooting the transition. Status simply indicates if the transition at that time is enable to shoot or not, while Priority indicates the degree of priority, as a natural number, if it is in conflict with another transition. In addition you have the option to change the network in a stochastic network with the option Exponential Time timeout that associates as a random number generated by the exponential distribution with the average value indicated in Time.

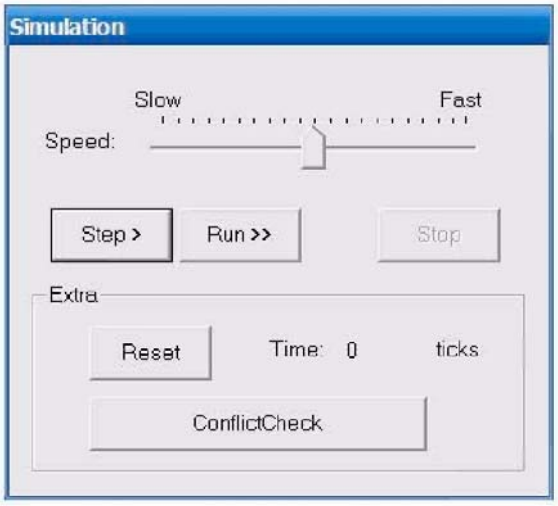
You can also upload from a file the time of transition with or without ACK clicking the option Load Time With ACK From File or Load Time From File.



Finally if we choose arc you have this options: Weight, Status and Probability. Weight obviously indicates the weight of the arc; it indicates the number of arcs which start at one place and go to a transition. Status is similar to transitions and Probability indicates the probability of choosing that path in the event of conflict. You can create also the inhibitor arc with the option Inibitor Arc, you can also load the files of time with the option Load Time With ACK From File.

The other two dialog windows, Places and Transitions, respectively indicating the lists of all places and all transitions. In the Place dialog window there are the ID and Name and also the number of tokens and the transition that they are input (Transition ID). In the Transition dialog window is indicated, ID, Name and places of entry and exit (PlaceIN) (PlaceOut) and the shutter time.

Let's see how you run the simulation. Once the model is created and initialized with the token and the other variables that we want to measure, it is ready to be analyzed by us. The Simulation window looks like this:



To make the best vision of simulation by the user you can adjust the speed of animation.

Also we have added the buttons Conflick Check which colours blue arcs that during the simulation are involved in the conflict and resolve the conflict in accordance with the characteristics of the designed model. You can also, at will, reset the time that is elapsed whenever you want to start again thanks to the simulation Reset button.

We conclude by outlining the features of the last button and the latest utility of Loki. The functions are: the view of the conflict (Check Conflict), the view average values (Results) and the ability to analyse the time between two transitions with the creation of a special arc (Fire2Fire).

The first, Conflict Check, is used before making a simulation of a model already designed. Once the button is pressed a table displays with all the places that generate conflict, the kind of conflict resolution and transitions with which they bind. In addition, the places with conflict are coloured blue and, by right clicking, we can interact by choosing the type of conflict resolution. This choice may be made with the property only after pressing the button Conflict Check from toolbar.

The other button is very simple and should be used after the simulation, it shows the average number of steps that have been performed during the simulation, in the places and transitions that were chosen for analysis with the option View Mean Value from Properties window.

Finally, we can create a special arc with the last button. The procedure for the creation of this arc is the same as that already seen, only his final appearance is different because it appears as a grey dashed line between two transitions. The results can be viewed only after the start of the simulation. The result can be seen by double-click on the arc that will show a window.

V. APPLICATION EXAMPLE

The term "performance evaluation" means a set of activities aimed to determining the

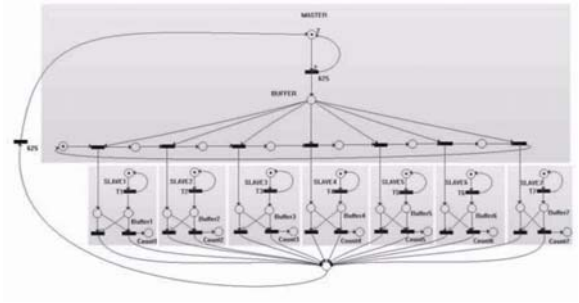
characteristics of a system on the basis of its behavior. Through the performance evaluation of a system you can know what it can offer under certain operating conditions, in order to delimit the application area where it can be used with success.

We will model and simulate a model for Bluetooth characterized by 7 Slave through the use of Petri nets, and finally we will analyze performance in various scenarios.

Among all the indices performance, we have decided to analyze the most interest: throughput. It shows the transmission capacity of a channel, which means the number of frames transmitted (token) in unit of time.

The purpose of this work is to create a Petri network that simulates, in its functionality, data transfer phase in a piconet in Bluetooth protocol, verifying the validity of performance in terms of throughput depending on the workload obtained by Loki simulator with actual protocol examined.

To make the various simulations, initially we have constructed the model with Networks Petri having 7 Slave. That, as it was demonstrated by the tests carried out during the creation, models well Bluetooth functionality to be analyzed in this experience. [4]



In the model, the master is characterized by an initial transition (lasting 625 μs), which is active in even slots or to the start of simulation, or after receiving a message from Slave (region A), the token is then sent to a slave, chosen second mode "round-robin", represented by output ring from the master himself (region B).

A generic Slave i (i = 1, ..., 7) consists of a generator of packages, which creates, through a transition with deterministic distribution (delay Ti), the tokens to be sent through an output buffer and a counter of submitted token. When the turn of the Slave is taken into consideration, if they are in their token buffer, a token is transmitted, its counter will be increased and control will be returned to the master; however, if the buffer is empty, a transition will be activated that will return only the control to the master (Region C).

This situation of choice was modeled using inhibitors arcs, which allows a transition to shoot only if in the place of entry there are any tokens.

Finally, to simplify the structure of the model, it was used a single transition (625 μs) from all slaves to the master (Region D) (in this way we haven't 14 separated transitions) to obtain a model easier to understand.

The purpose of the simulations is to measure the throughput (token / s) of each Slave to change of workload (token / s)

All simulations are characterized by a duration of 5 seconds. The times of generation of tokens are the same for all Slave.

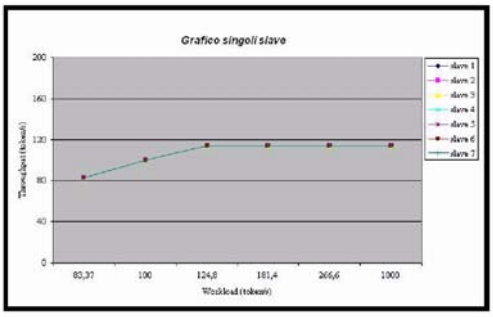
The tests were repeated by varying the delay T of the transitions, in a range of approximately 1 ms and 12 ms. This range was chosen from a situation of low load (12 ms) to achieve with its progressive changes, situations of saturation (1 ms).

From simulations, we obtained the following values:

#	T_g (s)	Token transmitted						
		Slave 1	Slave 2	Slave 3	Slave 4	Slave 5	Slave 6	Slave 7
1	12000	416	416	416	415	416	416	416
2	10000	499	499	499	499	499	499	499
3	8000	570	570	570	570	570	570	570
4	5500	570	570	570	570	570	571	571
5	3750	571	571	571	571	571	571	571
6	1000	571	572	572	571	571	571	571

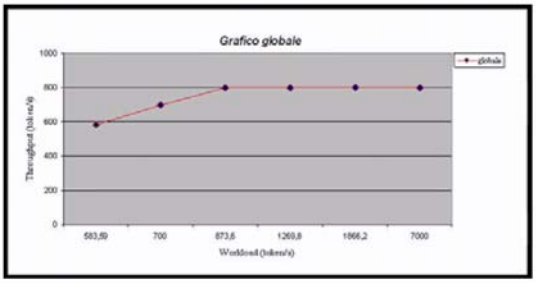
Results obtained for individual Slave

#	Workload (token/s)	Throughput (token/s)						
		Slave 1	Slave 2	Slave 3	Slave 4	Slave 5	Slave 6	Slave 7
1	83,37	83,2	83,2	83,2	83	83,2	83,2	83,2
2	100	99,8	99,8	99,8	99,8	99,8	99,8	99,8
3	124,8	114	114	114	114	114	114	114
4	181,4	114	114	114	114	114	114,2	114,2
5	266,6	114,2	114,2	114,2	114,2	114,2	114,2	114,2
6	1000	114,2	114,4	114,4	114,2	114,2	114,2	114,2



overall results obtained

#	Workload (token/s)	Throughput (token/s)	Token transmitted
1	583,59	582,2	2911
2	700	698,6	3493
3	873,6	798	3990
4	1269,8	798,4	3992
5	1866,2	799,4	3997
6	7000	799,8	3999



As you can see from global chart, throughput increases in a linear way with the workload until you

reach the saturation point (800 token / s), and from that point it begins to have a constant trend.

As we can expect, charts show a trend in which the values accurately reflect the theoretical basis of Bluetooth, because they are based on very specific conditions. Because you have 1600 time slots per second, half of which are reserved for the transmission of the master, there were 800 to be shared to all Slave. Because we have used a deterministic distribution for the creation of tokens output from slaves, charts relating to each of them are perfectly overlap.

VI. CONCLUSION

Ultimately the development of this Petri Networks simulator, Loki++, has allowed us to understand the importance of simulation in the study of Processes and networks.

A reliable tool for testing the complexity of modern networks and especially the wide variety of services offered by them is of great importance.

Thanks to the simulation of networks and related anomalies, it is possible to reproduce work environments that are increasingly similar to the networks in operation

REFERENCES

[1] Petri C A., *General Net Theory,* Proceedings of the Joint IBM & Newcastle upon Tyne Seminar on Computer Systems Design, 1976 J.

[2] Peterson J. L., *Petri Net Theory and the Modeling of Systems,* Prentice Hall, New Jersey, 1981

[3] Christoph Lindemann, *Performance Modelling with Deterministic and Stochostic Petri Nets,* John Wiley & Sons, Inc., 1998[4] McDermott-Wells, *What is Bluetooth?,* P. Potentials, IEEE, 2004