Internet Host Reliability Modeling with Time Petri Nets

Ali M. Meligy[1] Hani M. Ibrahim[2] Amal M. Aqlan[3]
Professor of Computer Science Lecturer of Computer Science PhD candidate
[1, 2, 3] Department of Mathematics, Faculty of Science, Menoufia University

ABSTRACT
Understanding the reliability of components is important for modeling the reliability of a distributed system. Careful modeling allows the construction of highly fault-tolerant distributed data applications and less expensive. Petri nets have been extensively used in the modeling and analysis of concurrent and distributed systems. Particular importance of Petri nets are in the development of concurrent and distributed systems. One of the standard concepts of time-dependent Petri nets is the one, where each transition gets a continuous time interval, specifying the range of the transition's reaction time. This extension of classical Petri nets is called Time Petri nets. In the internet the host is to obtain the data required to implement the functions of the Internet. In this paper, a method to model the internet host reliability with Time Petri nets is proposed.

Key words
Petri nets (PNs), Time Petri nets (TPN), Internet host, Reliability, Distributed System (DS).

1. INTRODUCTION
Petri net (PN) was used as a general purpose graphical and mathematical tool to describe the casual relationship between conditions and events in describing event-driven systems, i.e. discrete event (dynamic) systems. These systems may be asynchronous, contain sequential and concurrent operations, and involve conflicts, mutual exclusion and non-determinism [1].

Classical Petri nets have been extended to handle time. This results in two classes: Timed Petri nets and Time Petri nets. The timed Petri nets are derived by associating finite time duration with a transition or place, and the classic firing rule is modified to account for time. In a Time Petri net, the transition or place has one time interval with two bounds of time. When the time interval is associated with transitions, the first bound denotes the minimal time that must elapse, starting from the time at which the transition is enabled until this transition can fire. The second bound represents the maximum time during which that the transition can be enabled, and before which the transition must fire. The time Petri nets are more general than the timed ones since a timed Petri net can be modeled using a Time Petri net, but the converse is not true [2, 3, 4].

The Internet is a good example for distributed systems [5]. With the rapid growth of the World Wide Web and increased reliance on the web for almost every aspect of man's life today, Internet reliability is perhaps the most important challenge that researchers and practitioners face today. The reliability issues of the World Wide Web stem from various underlying factors [6].

This paper proposes the extension of Petri nets to deal with timing issues. A special advantage of Petri nets is their graphical notation, which reduces Petri nets learning time and simplifies their use. The primary purpose here is to design a clear example of the work of the host and provide an easy way to understand the model of information flow. However, some improvements can be added in order to make PN more powerful and flexible, for example, Time Petri nets.

The Internet is completely unreliable, then how we deal with that? So, the main goal of this paper is to propose a method the internet host reliability. The reset of this paper is organized as follows. After reviewing some related work in section 2. The basic concepts of Petri net, Time Petri net and distributed system are introduced in section 3, section 4, and section 5, respectively. The internet host reliability is presented in section 6. The Time Petri nets for Internet reliability is proposed in section 7.

2. RELATED WORK
Some efforts on reliability modeling using Petri nets can be summarized as follows. Bobbio, *et al.* use the generalized stochastic Petri net (GSPN) to support system dependability analysis [7]. Their approach involves converting fault trees into a GSPN model for the purpose of obtaining both qualitative and quantitative analysis results for the modeled system. Everdij and Blom develop piecewise deterministic Markov processes (PDP) models using dynamically colored Petri nets (DCPN) [8]. Petri nets are also applied in safety analysis of a system as shown by Leveson and Stolzy, where Petri nets are used to design and analyze the safety and fault tolerance of a system [9]. Using timed Petri nets, they prove that paths to high risk states can be removed based on reachability analysis. Buy and Sloan propose a method to automatically analyze the timing properties of concurrent systems [10]. Their method uses simple time Petri nets to analyze concurrent software systems developed in Ada. There are many previous efforts for formal modeling and analysis of various systems using Petri nets [11, 12, 13, 14, 15].

3. THE BASIC CONCEPTS OF PETRI NET
PN is a graphical and mathematical modeling tool which possesses the advantages of graphical notation and simple semantics [16, 17, 18, 19, 20, 21]. Fig. 1 shows an example of PN.

A PN is a 5-tuple (P, T, A, W, M_0), Where

1) $P = \{p_1, p_2, \cdots, p_m\}$ is a finite set of places. A place represents a circle, such as, p_1, p_2 and p_3 in Fig. 1.
2) $T = \{t_1, t_2, \cdots, t_n\}$ is a finite set of transitions. A transition represents a bar, such as t_1 in Fig. 1. $P \cap T = \emptyset$; and $P \cup T \neq \emptyset$.
3) $A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs connecting places and transitions, such as the arrowhead from p_1 to t_1 depicted in Fig. 1.

4) $W: A \rightarrow \{1,2,3,\cdots\}$ is a weight function, whose weight value is positive integers. Arcs, i.e., arrowhead, are labeled with weights. For example, in Fig. 1, the arrowhead from t_1 to p_3, which is labeled with ''2", is denoted as $W(t_1,p_3) = 2$. When the weight is unity and/or ''1", the label of arc is usually omitted, e.g., $W(p_1,t_1) = 1$ is omitted in Fig. 1.

5) $M_0: P \rightarrow \{0,1,2,\cdots\}$ is the initial marking. If there are k tokens inside place p_i, it is said that p_i is marked with k tokens. For example, in Fig. 1a, p_1 is marked with one token, which is denoted as $M(p_1) = 1$, p_2 is marked with two tokens, which is denoted as $M(p_2) = 2$. If Fig. 1a is the initial status, the initial marking is denoted as $M_0(p_1,p_2,p_3) = \{1,2,0\}$.

A transition t is said to be fired if all its input places p_i are marked with at least $W(p_i,t)$ tokens, where $W(p_i,t)$ is called the firing condition of transition t. For example, in Fig. 1, the firing conditions of t_1 are $W(p_1,t_1) = 1$ and $W(p_2,t_1) = 2$.

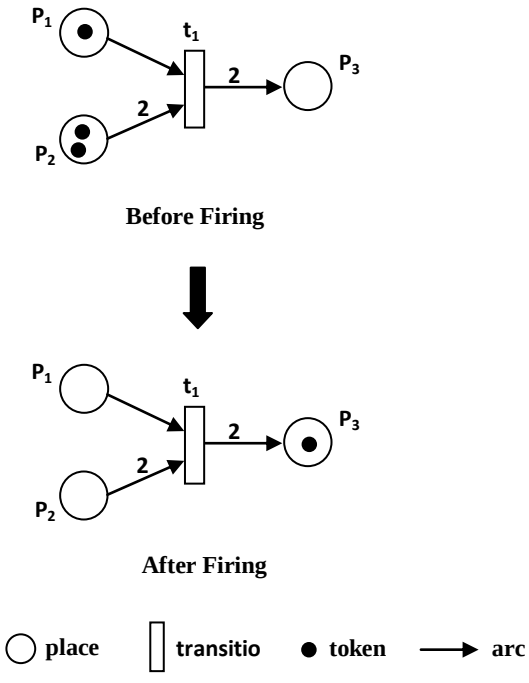

Fig. 1: An example of Petri Nets.

4. TIME PETRI NETS

Merlin's Time Petri nets (*TPNs*) extend Petri nets with temporal intervals associated with transitions, specifying firing delays for transitions [22, 23].

A Time Petri Net is $Z = (P,T,A,W,M_0,I)$ where:

1. $S(Z) = (P,T,A,W,M_0)$ is a Petri Nets.
2. $I: T \rightarrow Q_0^+ \times (Q_0^+ \cup \{\infty\})$ *and* $I_1(t) \leq I_2(t)$ for each $t \in T$, where $I(t) = (I_1(t), I_2(t))$.

The Petri net $S(Z)$ is referred to as the skeleton of Z. Q is the set of positive rational numbers. I is the interval Function of Z, $I_1(t)$ and $I_2(t)$ the *earliest firing time* of t $(eft(t))$ and the *latest firing time* of t $(lft(t))$, respectively.

The firing interval I of some transitions may be equal to *zero*, which means that these firings are instantaneous; all such transitions are called *immediate*, while the others are called *timed* transitions.

In *TPN*, the enabling condition of a transition is the same as in Petri Nets. According to the definition of *TPN* the "*enabledness condition*" will be:

$$(\forall p)(M(p) \geq A(t,p)) \quad (1)$$

Some transitions may be enabled by a marking M, but not all of them may be allowed to fire due to the firing constraints of transitions($eft's$ and $lft's$). So the "*firability condition*" depends of two conditions:

1. The transition is enabled, formally expressed by *Eq*(1).
2. Expresses the fact that enabled transitions may not be fired before its eft and must be fired before or at its lft unless another transition fires before, modifying marking M so the transition is no longer enabled.

According to the second condition, a transition t_i enabled by M at absolute time τ could be fired at the firing time θ, iff θ is not smaller than the eft of transition t_i and not greater than the smallest of the $lft's$ of all the transitions enabled by marking M, that is: eft of $t_i \leq \theta \leq \min\{lft\ of\ t_k\}$ where k ranges over the set of transitions enabled by M. If transition t_i fires, it leads the system to another state, at time $\tau + \theta$.

5. DISTRIBUTED SYSTEMS

We can define a distributed system as a collection of loosely coupled processors interconnected by a communication network. The processors (sites, nodes, computers, machines, or hosts) in a distributed system may vary in size and function. They may include small microprocessors, workstations, minicomputers, and large general-purpose computer systems. The purpose of the distributed system is to provide an efficient and convenient environment for such sharing of resources [24].

The main advantages of distributed systems are [25]

1. **Resource Sharing:** In distributed systems, resource sharing provides mechanisms for sharing files at remote sites, processing information in a distributed database, printing files at remote sites, and using remote specialized hardware devices.
2. **Reliability and Availability:** In a distributed system, the failure of a few components of the system may not affect the overall system availability.
3. **Communication**: The users of a distributed system have the possibility of exchanging information. Using distributed systems people working at distant

positions are able to perform common work in a powerful way.

The internet is a very large distributed system. It is designed exemplify the wide range of services and applications that are supported by computer networks and to being the discussion of the technical issues that underlie their implementation. The internet is a vast interconnected collection of computer networks of many different types. Programs running on the computers connected to it interact by passing messages, employing a common means of communication. The design and construction of the internet communication mechanisms (the internet protocol) is a major technical achievement, enabling a program running anywhere to address messages to programs anywhere else [5].

6. INTERNET HOST RELIABILITY

An Internet host can be a machine or an application connected to the Internet that has an Internet Protocol address (*IP* address). An *IP* address is used to uniquely identify every host on the Internet. Because these *IP* address are unique, they can be used as the source and destination *IP* addresses in any *IP*-based communication. At the most basic level, Internet hosts fall into two categories, Servers and Clients [26].

Reliability is the probability that a system will remain constantly available over a fixed time period. Reliability is usually defined as the probability of successful operation of a mission under predefined operating conditions and for a specified mission time. Consider a system that functions for one second, then fails, but recovers in a small number of milliseconds. This system would have a high availability, but would not be useful for applications like process control that must remain continuously functioning for extended periods. Reliability is a more appropriate measure for these applications because it includes duration [27].

An accurate estimate of host reliability is important for correct analysis of many distributed systems. We estimated host system reliability by querying a large number of hosts to find how long they had been functioning.

7. THE PROPOSED TIME PETRI NET

Data were collected from as many hosts as was feasible using only data that could be obtained via the internet with no special privileges or added monitoring facilities. This was principally done by polling host using *Sun RPC* [28] to query rpc.statd to obtain up-times and to test availability, and by querying domain servers [29] to obtain host-specific information [30].

Time Petri nets constitute general models for time dependent systems. In the following, time transitions are assumed to exist in a time Petri net. Normally, a time interval is specified by two bounds of time: a *maximum* time a and a *minimal* time b. The time interval $[a, b]$ is designed by real numbers, but the interval bounds are nonnegative rational, integer or natural numbers. In this paper we will use a nonnegative real numbers.

First, consider the following:

- We choose a number of hosts, e.g. 100 hosts.
- We identify a common time period for the hosts. For example, two months as the period is [0.1, 6], where a = 0.1 and b =6 is the days and the increase is by 0.1 at each step.

In Fig. 2 we have two types of transition: timed transition and immediate transition. t_1 is timed transition while from t_2 to t_8 and t_0 are Immediate transitions. As soon as becoming firing, immediate transitions will be fired without delay.

Consider the Time Petri net in Fig. 2:

- Time interval [0.1, 6] is associated with transitiont_1. Obviously, transition t_1 is enabled at time τ = 0 and a token is reserved in place p_1 in order to fire transitiont_1. From the semantics of Time Petri nets, transition t_1 can be fired after time θ = 0.1, and must be fired before time θ = 6. Suppose that each transition works in the earliest firing mode, i.e., each enabled transition is fired as soon as its minimal time elapses. Then, at time θ = 0.1 transition t_1 is fired, and token is put into p_2. When t_2 is firing, this is done directly to the lack of time constraints.
- At time τ = 0.1, transition t_1 is enabled. At time θ = 0.2, transition t_1 is fired again.
- At time τ = 0.2, transition t_1 is enabled. At time θ = 0.3, transition t_1 is fired. Thus, to be completed on time.

The main work of a host is to process requests from client and respond with the correct information. Fig. 2 is the *TPN* graph modeling the described behavior to internet host reliability. This is as follows:

1-Start with p_1, where a token in p_1 which means the requested message, where the message consists of single datagram message sent from the host to the destination. p_1 is buffer as well as $p_2, \cdots, p_{10}$.

2-The p_2 receives the requested message across the t_1, which means Polling host protocol that sends the request message.

3-t_2 distribution of the requested message to p_3 and p_4, using *Sun RPC* Which provides two services are:

a. t_3 performs the *rpc.statd* on the requested message and store the resulting information (up-time to test availability) in p_5.

b. t_4 performs the *domain servers* on the requested message and store the resulting information (host specific information) in p_6.

4-t_5 collects the required data from p_5 and p_6 then placed in p_7 after their processing.

5-t_6 examines the data and put the result in p_8. If "Yes" go to step 6, otherwise go to step 7 (Yes means that the data required in full. No mean that the required data is incomplete).

6-t_7 receives answer “Yes” and put data in p_9. Then stop.

7-t_8 receives answer "No" and put the requested message again in p_{10}.

8-t_0 receives the requested message again and go to step 1.

This process (the previous steps) in less than one second, the process is repeated until the end of time means at time $b = 6$. The period for each of the t_2 to t_8 is [0, 0] until the transition to the next step without delay.

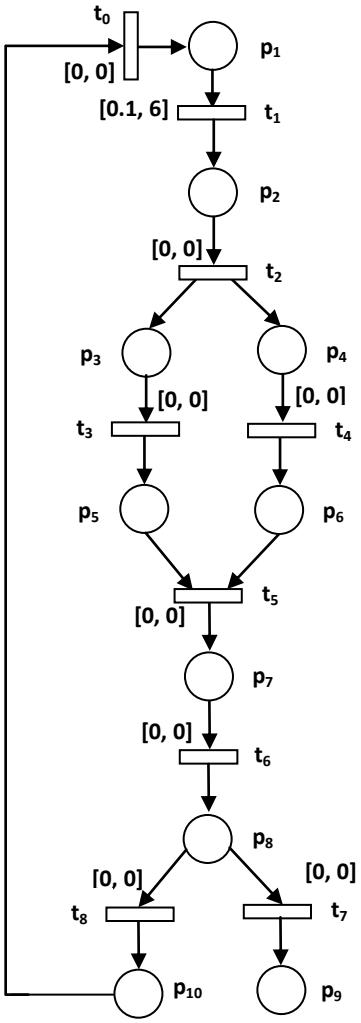

Fig 2: TPN graph for acquiring information about internet host reliability

Compared to the results of Long et. al. [30] we have a more compact graphical model using time Petri nets. This model may be mathematically batter analyzed.

Our model consider as a first step in the PN modeling of reliability. In further, we intend to model the reliability of the Internet host using other types of Higher-order Petri nets.

8. CONCLUSION

In this paper, we discussed the applicability of Time Petri Nets to the modeling of internet host reliability. Petri nets, as a graphical and mathematical tool, provide a uniform environment for modeling, analysis and design of distributed systems. One of the major advantages of using Petri nets is that the same methodology can be used for the modeling, planning and scheduling, and system design in the internet host reliability. In this paper the Time Petri nets for modeling the internet host reliability is proposed. This method use transitions to identify the sequences of the events in the Time Petri nets while places to identify the sequences of the states. Further, we also discussed the functionality of the proposed method is presented.

9. REFERENCE

[1] Gu . T., and Bahri P.A. 2002. A Survey of Petri Net Applications in Batch Processes, computers in industry, vol.47, pp 99-111.

[2] Ramchandani C. 1974. Analysis of Asynchronous Concurrent Systems by Timed Petri Nets. Massachusetts Inst. Tech. Rep.120.

[3] Merlin P.M., and Farber D.J. 1976. Recoverability of Communication Protocols -Implications of a Theoretical Study. IEEE Trans. Commun., vol. 24 (9), pp 1036–1043.

[4] Gu T., and Dong R. 2005. A Novel Continuous Model to Approximate Time Petri Nets: Modeling and Analysis. Int. J. Appl. Math. Comput. Sci., vol. 15(1), pp. 141–150.

[5] Couloris G., Dollimore J., and Kinberg T. 2001. *Distributed Systems- Concepts and Design*. 4th Edition, Addison-Wesley, Pearson Education, UK.

[6] Peiravi A. 2010. Application of String Matching in Internet Security and Reliability. Journal of American Science, vol. 6(1), pp. 25-33.

[7] Bobbio A., Franceschinis G., Portinale L., and Gaeta R., 1999. Exploiting Petri nets to support fault-tree based dependability analysis. In Proc. 8th Int. Workshop on Petri Nets and Performance Models, Zaragoza, Spain, pp. 146-155.

[8] Everdij M. and Blom H. 2003. Petri-nets and hybrid-state Markov processes in a power-hierarchy of dependability models. In Proc. IFAC Conf. Analysis and Design of Hybrid Systems, Saint-Malo, Brittany, France.

[9] Leveson N. G., and Stolzy J. L. 1987. Safety analysis using Petri nets. IEEE Trans. Softw. Eng., vol. 13(3), pp. 386-397, Mar.

[10] Buy U. and Sloan R. 1993. A Petri net-based approach to real-time program analysis. In Proc. 7th Int. Workshop on Software Specification and Design, Redondo Beach, California, pp. 56-60.

[11] Wang H. and Zeng Q. 2008. Modeling and analysis for workflow constrained by resources and nondetermined time: an approach based on Petri nets. IEEE Trans. Syst., Man and Cybern. A, Syst., Humans, vol. 38(4), pp. 802-817.

[12] Lee J.-S., Zhou M. C., and Hsu P.-L. 2008. Multiparadigm modeling for hybrid dynamic systems using a Petri net framework. IEEE Trans. Syst., Man and Cybern. A, Syst., Humans, vol. 38(2), pp. 493-498.

[13] Ma L., and Tsai J. J. P. 2008. Formal modeling and analysis of a secure mobile-agent system. IEEE Trans. Syst., Man and Cybern. A, Syst., Humans, vol. 38(1), pp. 180-196.

[14] Shen V. R. L., and Juang T. T.-Y. 2008. Verification of knowledge-based systems using predicate/transition nets. IEEE Trans. Syst., Man and Cybern. A, Syst., Humans, vol. 38 (1), pp. 78-87.

[15] Du Y. Y., Jiang C. J., and Zhou M. C. 2007. Modeling and analysis of real-time cooperative systems using Petri nets. IEEE Trans. Syst., Man and Cybern. A, Syst., Humans, vol. 37(5), pp. 643-654.

[16] Peterson J. L. 1977. Petri Nets. ACM Computing Surveys (CSUR), vol. 9(3), pp 223–252.

[17] Peterson J. L. 1981. Petri Net Theory and the Modeling of Systems. Prentice Hall, Englewood Cliffs.

[18] Dwyer M. B., Clarke L. A. 1996. A Compact Petri Net Representation and its Implications for Analysis. IEEE Trans.on Software Engineering, vol. 22(11), pp. 794–811.

[19] Koriem S. M. 2002. A Fuzzy Petri Net tool for Modeling and Verification of Knowledge-based Systems. The Computer Journal, vol. 43(3), pp. 206–223.

[20] Murata T. 1989. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, vol. 77, pp. 541–580.

[21] Chang Y.-C., Huang Y.-C., and Chu C.-P. 2009. B^2 Model: A Browsing Behavior Model based on High-Level Petri Nets to Generate Behavioral Patterns for e-learning. Expert Systems with Applications, vol. 36, pp 12423–12440.

[22] Peter Bachmann J., and Popova-Zeugmann L. 2010. Time-independent Liveness in Time Petri Nets. IOS Press , Fundamenta Informaticae, vol. 101, pp. 1–17.

[23] Gonzalez P. M., and Silva J. R. 2008. Using Time Petri Nets for Modeling and Verification of Timed Constrained Workflow Systems. Abcm Symposium Series in Mechatronics, vol. 3 - pp.471-478.

[24] Silberschatz I., Galvin P. B., and Gagne G. 2001. *Operating System Concepts*. 6th Edition, Wiley, John & Sons.

[25] Falai L. 2007. Observing, Monitoring and Evaluating Distributed Systems. Ph.D. thesis, Universitµa degli Studi di Firenze.

[26] "What is an Internet Host? "http://www.inetdaemon.com/tutorials/internet/ip/whatis_ip_host.shtml".

[27] Long D., Muir A., and Golding R. 1995. A Longitudinal Survey of Internet Host Reliability. In Proc. of the Symposium on Reliable Distributed Systems.

[28] Sun Microsystems, Incorporated. RPC: Remote Procedure Call Protocol Specification Version 2. Tech. Rep. RFC-1057, USC Information Sciences Institute, June, 1988.

[29] Mockapetris P.V. 1987. Domain Names-Concepts and Facilities. Tech. Rep. RFC-1034, USC Information Sciences Institute.

[30] Long D., Carroll J., and Park C. 1991. A study of the Reliability of Internet Sites. In Proceedings of the Tenth Symposium on Reliable Distributed Systems, pp. 177–186, Pisa.