TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 31/49 pp175-179
Volume 12, Number S1, July 2007

A Novel Formal Analysis Method of Network Survivability Based on Stochastic Process Algebra*

ZHAO Guosheng (赵国生)[1,2,**], WANG Huiqiang (王慧强)[2], WANG Jian (王 健)[2]

1. Center of Computer Network and Information, Harbin Normal University, Harbin 150001, China;
2. Institute of Computer Science and Technology, Harbin Engineer University, Harbin 150001, China

Abstract: Stochastic process algebras have been proposed as compositional specification formalisms for performance models. A formal analysis method of survivable network was proposed based on stochastic process algebra, which incorporates formal modeling into performance analysis perfectly, and then various performance parameters of survivable network can be simultaneously obtained after formal modeling. The formal description with process expression to the survivable network system was carried out based on the simply introduced syntax and operational semantics of stochastic process algebra. Then PEPA workbench tool was used to obtain the probability of system's steady state availability and transient state availability. Simulation experiments show the effectiveness and feasibility of the developed method.

Key words: formal analysis; stochastic process algebra; network survivability; performance analysis

Introduction

At present, survivability has been a new research direction of network security technology. The definitions of survivability have been introduced by previous researchers[1,2], which define survivability as the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents. The system is in the broadest possible sense, including networks and large-scale systems[3]. In the area of survivability, the formal modeling technology is the most important foundational work, which may carry out the precise description and formal analysis for the dynamic behavior of network system through the standard formal language. The final goal of formal modeling is to perform the qualitative and quantitative analysis for the network system described. However, so far this field is being at the exploration stage. The existing most formal modeling methods aim at local network or specific system[4]. The paradigm that can simultaneously unify the qualitative and quantitative analysis into the formal modeling has not been found in existing research.

The European reliability research group[5] firstly remarked on the facet of survivability formalization, starting the 'X-key application' project. Although the 'X' may be one all-embracing characteristic, the formal method developed by this project was only applied in the fault-tolerant field until now. Westmark[6] proposed a formal description template of survivability from the angle of the smallest rank of service capability provided by network system in face of threat. Knight and Sullivan[7] gave a kind of survivable formalization description with sextuple, which is not precise enough to support an engineering approach to the survivability analysis of actual network system. Ellison[8] took service as the center, descriptively defined the survivability using finite state machine (FSM). Park and Chandramohan[9] introduced UML language to carry on the formal description of the survivability. Koroma and Li[10] introduced Semi-Markov process to formalize the survivability, but he only defined the paradigm

Received: 2007-02-01

* Supported by the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20050217007)

** To whom correspondence should be addressed.
E-mail: zhaoguosheng@hrbeu.edu.cn; Tel: 86-451-88060155

system not the abstract system. Above several typical formal modeling methods mostly belong to conceptual description, which are difficult to be applied in the qualitative or quantitative analysis of survivability if they do not have a great many following work to do.

In the analysis of survivability, the existing literatures are mostly either qualitative or quantitative analysis. CMU/SEI proposed a representative SSA[11] method, which firstly divided the system into a security nucleus that cannot be broken and a restorable part, in view of certain attack pattern, gave the corresponding resistance, recognition, and recovery strategy, but it is merely a qualitative analysis method. Moitra and Konda[12] have developed a set of simulation models, which may quantitatively analyze the defense mechanisms and viability of various networks. Kring[13] proposed a kind of analysis framework that can converts the question of survivability analysis into a typical graph question based on the question space transformation idea, but it is a kind of qualitative analysis yet. By viewing service as AND-OR structure，Guo and Ma[14] presented a quantitative analysis method for services survivability based on the concept of configuration.

Because the high-level stochastic modeling methods have perfect performances in the description of system behavior relations and algebraic characters[15], which can also integrate quantitative and qualitative analysis into formal description, they have been gradually widely applied in network security analysis. This paper firstly uses the high-level stochastic formal modeling method to formally model and analyze the survivability of network system.

1 Stochastic Process Algebra (SPA)

The syntax of SPA component, P, is represented as

$$P ::= \text{Nil} \mid (a,\lambda).P \mid P+Q \mid P \|_L Q \mid P/L \mid A \qquad (1)$$

Prefix, $(a,\lambda).P$: This represents the process P becomes a new one ofter an action, a. The time taken to perform a is described by an exponentially distributed random variable with parameter λ. The rate parameter may also take the value Nil, which makes the action passive in a cooperation.

Choice, P+Q: A race is entered into between components P and Q. If P evolves first, then any behavior of Q is discarded and vice-versa. This is often called competitive choice.

Hiding, P/L: Actions in the set L that emanate from the component P are rewritten as silent Γ-actions. The actions in L can no longer be used in cooperation with other components.

Constant, A: It is convenient to be able to assign names to patterns of behavior associated with components. Constants are components whose meaning is given by a defining equation. For example, $A \overset{\text{def}}{=} (a,r)$. A performs a at rate r forever.

Cooperation, $P \|_L Q$: P and Q run in parallel and synchronies over the set of actions in the set S. If P is to evolve with an action $a \in S$, it must first wait for Q to reach a point where it is also capable of producing an a-action, and vice-versa. In cooperation, the two components then jointly produce an a-action with a rate that reflects the slower of the two components.

2 Formal Modeling

One of the main advantages of stochastic process algebras is that it allows the creation of highly modular model descriptions. It may also accurately describe process behaviors and mutual relationships among system modules. Obviously, using stochastic process algebras to formalize the survivable network system is suitable. The structure of survivable network system usually involves a service request module and a server module that provides different services to relevant users as redundant way. So, in our research, we can consider our survivable system as the parallel composition of two components or processes (as shown in Fig. 1). System initial state can be illuminated as

$$\text{System} := \text{Request} \|_A \text{Server} \qquad (2)$$

where $A := \{\text{key\_job,non\_key\_job,fail}\}$.

The process request represents the system loads caused by the different service quest, which the legal users and the attackers claim. The mainframe itself is modeled by the server process.

2.1 Request module modeling

Referring to the definition of survivability described by CMU/SET[3], we divided the service requests submitted to system into the key service request and the non-key service request. Moreover, we also take the attack request as a part of request module. So, the request module may be described as

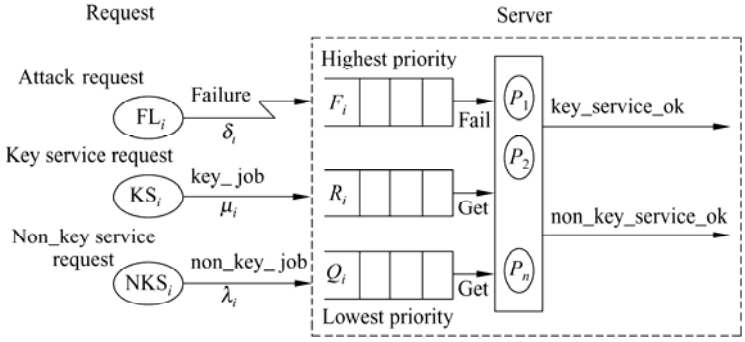

Fig. 1 Model structure

$$\text{Request} := \text{Non\_key\_service\_request}_1 \|_{\{d\}} \text{Key\_service\_request}_1 \|_{\{d\}} \text{Attack\_request}_1 \quad (3)$$

In practical network running, the service request rates are changed along with the change of network real performance, so we use three synchronous process flows to model each service request. In normal state, three kinds of service requests' speed are $\delta_1, \mu_1, \lambda_1$, respectively, but in busy phase, are $\delta_2, \mu_2, \lambda_2$, respectively. The Request module could be obtained as follows:

$$\begin{aligned}&\text{Key\_service\_request}_1 := (c, \phi).\text{Key\_service\_request}_2 + (\text{key\_job}, \mu_1).\text{Key\_service\_request}_1,\\ &\text{Key\_service\_request}_2 := (c, \phi).\text{Key\_service\_request}_3 \text{L} + (\text{key\_job}, \mu_2).\text{Key-service\_request}_2,\\ &\text{Key\_service\_request}_3 := (c, \phi).\text{Key\_service\_request}_1 \text{M}\end{aligned} \quad (4)$$

The processes $\text{Key\_service\_request}_i$ and $\text{Non\_key\_service\_request}_t$ generate repeatedly after an exponentially distributed time delay with the rates μ_i and λ_i respectively. By analogy, the process Attack_request_i generates with rate δ_i, which can be regarded as special services request that require all processors and possess preemptive priority over the other request classes. All request processes change their phase after an exponentially–ϕ distributed delay by executing action *c*, which however is not synchronized with the server module. In phase 3 all request processes are idle.

2.2 Server module modeling

The second main component of our model is server module, which include one queue and *n* processes unit. A queue for various requests has to wait until a processor becomes free. The processing unit is modeled by the parallel composition of *n* identical processes, each modeling one processor. Its formal description may be as follows:

$$\begin{aligned}&\text{Server} := \text{Queues} \|_B (P_1 \|_{\{\text{fail,repair}\}} P_2 \| \ldots \|_{\{\text{fair,repair}\}} P_n),\\ &\text{Queues} := F_0 \|_C (R_0 \|_{\{\text{get\_non\_key\_job}\}} Q_0),\\ &B = \{\text{get\_key\_job,get\_non\_key\_job, fail, repair}\}\end{aligned} \quad (5)$$

where $C = \{\text{non\_key\_job,get\_non\_key\_job, key\_job, get\_key\_job}\}$

2.2.1 Queues modeling

The queuing component is responsible for the buffering of incoming requests. Additionally, we integrated a scheduling mechanism in this component. Requests are processed according to an FIFO strategy with priorities. Non-key service requests have the lowest priority, while attack requests have the highest priority. To each class of requests a separate queue process is assigned. The queue Q_0 stores the low priority jobs while the queues R_0 and F_0 are for the key service requests and the attack requests respectively. The queues modeling could be obtained as formula (6).

The priority mechanism is realized by appropriate synchronization of the three queue processes. The process Q_i for instance is only able to deliver a non-key service request to a processor if the other two queues are in state R_0 and F_0. Otherwise, the action get_non_key_job is not enabled. Another important fact is that the process F_i prohibits the insertion of new requests if it is in state F_1, i.e., if a failure caused by one attack occurred. The mean duration of the actions get_non_key_job and get_key_job is $1/a$.

$$
\begin{aligned}
&Q_0 \coloneqq (\text{non\_key\_job},1).Q_1,\\
&Q_i \coloneqq (\text{non\_key\_job},1).Q_{i+1} + (\text{get\_non\_key\_job},a).Q_{i-1},\\
&Q_{l_1} \coloneqq (\text{get\_non\_key\_job},a).Q_{l_1-1},\\
&R_0 \coloneqq (\text{key\_job},1).R_1 + (\text{get\_non\_key\_job},1).R_0,\\
&R_i \coloneqq (\text{key\_job},1).R_{i+1} + (\text{get\_key\_job},b).R_{i-1},\\
&R_{l_2} \coloneqq (\text{get\_key\_job},b).R_{l_2-1},\\
&F_0 \coloneqq (\text{get\_non\_key\_job},1).F_0 + (\text{key\_job},1).F_0 +\\
&\quad (\text{get\_key\_job},1).F_0 + (\text{fail},1).F + (\text{non\_key\_job},1).F_0,\\
&F_1 \coloneqq (\text{repair},\beta).F_0
\end{aligned} \tag{6}
$$

2.2.2 Processing unit modeling

Each processor waits until it can carry out an action or until a failure occurs. As failures have preemptive priorities over the other two request classes, all processors stop processing if the failed action becomes enabled and have to enter recovery state P_f. The *i*-th processing unit module formal description is as follows:

$$
\begin{aligned}
&P_i \coloneqq (\text{get\_key\_job},1).P_{i0} + (\text{fail},1).P_{if} +\\
&\quad (\text{get\_non\_key\_job},1).P_{i1},\\
&P_{i0} \coloneqq (\text{key\_job\_ok},\eta).P_i + (\text{fail},1).P_{if},\\
&P_{i1} \coloneqq (\text{non\_key\_job\_ok},\xi).P_i + (\text{fail},1).P_{if},\\
&P_{if} \coloneqq (\text{repair},\beta/n).P_i
\end{aligned} \tag{7}
$$

3 Results and Analysis

We use the above formal model to realize the performance analysis of network survivability based on the analysis tool of PEPA Workbench[16]. Using the mathematic software package of Maple, this paper first dealt with the data file of transformation speed matrix Q corresponding to continuous time Markov chain (CTMC) which have been produced by PEPA workbench tool, and then realized the measures of system's stable state availability and the transient state availability.

We assumed the initial parameters as follows: $\lambda_1 = 0.016\ 67\ \text{min}^{-1}$, $\lambda_2 = 0.16\ \text{min}^{-1}$, $\mu_1 = 0.033\ \text{min}^{-1}$, $\mu_2 = 2\ \text{min}^{-1}$, $l_1 = 40$, $l_2 = 20$, $\delta_1 = 0.000\ 35\ \text{min}^{-1}$, $\phi = 0.003\ 34\ \text{min}^{-1}$, $n = 4$.

Figure 2 shows the repair rate and attack frequency has a large impact to system's steady state availability. Along with the increase of repair rate β, system's steady state availability enhanced. And while attack frequency δ_2 is smaller, the probability of system's steady state availability is bigger.

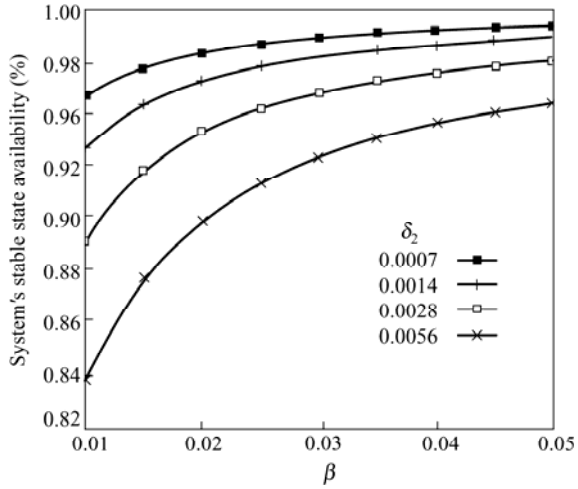

Fig. 2 System's steady state availability

In Fig. 3, the curve of $\beta = 0.01$ shows clearly the change of system's transient state availability along with *t*, and the size of β decided the probability of system's transient state availability. In initial stages of service, the system's availability drops because the failures occurred. However, the system subsequently switched to the repair state, system's transient state availability enhances. After $t = 1400$ min the system reached its steady state.

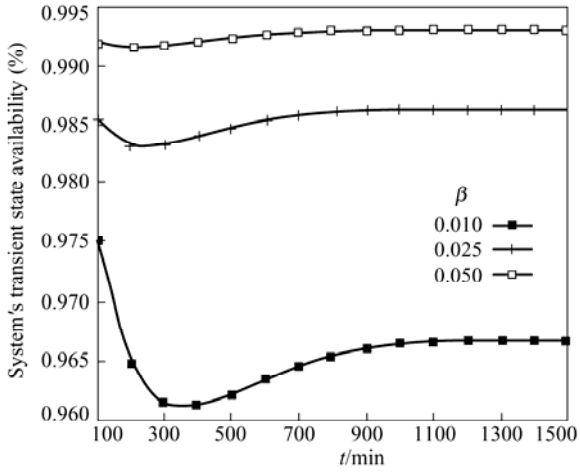

Fig. 3 System's transient state availability

4 Summary

The stochastic process algebra provides a powerful tool for the formal modeling and performance analysis of survivable network system. From formal modeling to analyze current various performance parameter of network system, the stochastic process algebra not only helps the accurately qualitative and quantitative

analysis for network viability, but also provides the reasonable judgment for the system's maintenance and dynamic configuration. Although this paper has done some foundational works in the formalization modeling and the performance analysis, the method proposed is still coarse at present. Up to now, we are not able to cope with the state spaces that contain more than 100 000 states if we use standard workstations for the analysis. Future works will focus on perfecting the formalization model proposed.

References

[1] Knight J C, Strunk E A, Sullivan K J. Towards a rigorous definition of information system survivability. In: Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03). New York: IEEE Press, 2003: 78-89.

[2] IEEE Standard 1061-1992. Standard for a Software Quality Metrics Methodology. New York: Institute of Electrical and Electronics Engineers, 1992.

[3] Lipson H F. Survivability—A new security paradigm for protecting highly distributed mission critical systems. CERT-Carnegie Mellon University, Pennsylvania United States, 2002, http://www.cert.org/archive/pdf/survivparadigm. pdf.

[4] Mcdermott J. Attack-potential-based survivability modeling for high-consequence systems. In: Proceedings of the 3rd IEEE International Workshop on Information Assurance (IWIA'05). CallegePark, Maryland, USA, 2005: 119-130.

[5] Lann G L. Predictability in critical systems. Lecture Notes in Computer Science.Formal Techniques in Real-Time and Fault-Tolerant System. Berlin: Springer-Verlag, 1998: 315-338.

[6] Westmark R. A definition for information system survivability. In: Proceeding of the 37th Hawaii Internal Conference on Systern Sciences (HICSS'04). New York: IEEE Press, 2004: 2086-2096.

[7] Knight J C. Sullivan K J. On the definition of survivability. University of Virginia, Department of Computer Science, Technical Report CS-TR-33-00, 2000, http://www.cs. virginia.edu/-jck/recentpapers.htm.

[8] Ellison R J. Fisher B D, Linger R, et al. Survivable network systems: An emerging discipline. Pittsburgh, Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI97-TR-013, 1997.

[9] Park J, Chandramohan P. Static vs. dynamic recovery models for survivable distributed systems. In: Proceedings of the 37th Hawaii International Conference on System Sciences. Washington, USA, 2004.

[10] Koroma J, Li W. A generalized model for network survivability. In: Proceedings of the 2003 Conference on Diversity in Computing (TAPIA'03). Atlanta, Georgia, USA, 2003: 47-51.

[11] Ellison R, Fisher D, Linger R. Survivable network system analysis: A case study. *Software*, *IEEE*, 1999, **16**(4): 70-77.

[12] Moitra S D, Konda S L. A simulation model for managing survivability networked information systems. Carnegie Mellon University, 2000, Technical Report CMV/SEI-2000-TR-020, 245-251.

[13] Kring W. A graph-based model for survivability applications. http://www.cs.uidaho.edu/krings/publications.html, 2005.

[14] Guo Y B, Ma J F. Quantifying survivability of services in distributed system. *Journal of Tongji University*, 2002, **30**(10): 1190-1193. (in Chinese)

[15] Lin C, Wang Y, Li Q L. Stochastic modeling and evaluation for network security. *Chinese Journal of Computers*, 2005, **28**(12): 1943-1956. (in Chinese)

[16] Clark G, Gimore S, Hillston J, et al. Experience with the PEPA performance modeling tools. *IEEE Software*, 1999, **146**(1): 11-19.