Petri-nets Based Availability Model of Fault-Tolerant Server System

Shi Jian Wang Shaoping Shang Yaoxing
School of Automation Science and Electrical Engineering
Beijing University of Aeronautics and Astronautics
Beijing, 100083, China
shijian123@sina.com shaopingwang@vip.sina.com syxstar@gmail.com

Abstract—**Fault-Tolerant capacity of the key server will effect integrality and restorability of the data in network system, so in order to increase system's availability, cluster technique has been adopted widely. In despite of achieving high system availability with cluster technique, it makes system very complexly and availability analysis very difficult. By analyzing redundant server system's structure and work process, stochastic Petri net method is adopted to model processing cell availability, data memory disc availability and server availability. Integrating above models, cluster system availability model can be obtained.**

Keywords—**redundant server system, availability model, cluster technique, stochastic Petri net.**

I. INTRODUCTION

In traditional analysis to the computer networks based on topological connectivity, it is generally assumed that the reliability of devices in computer network is definite, so the analysis process to equipment always is ignored. However in practice, most of devices such as switcher and server in communication network are fault dependent elements whose availability is not the simple product of reliabilities of hardware and software. The fault dependencies in these combined software and hardware system come from four aspects as follows.

- Functional dependency: According to the system requirements, it is necessary to establish the communication among hardware components and software modules that transmit, share and verify the data in computer network system.
- Structural dependency: In order to realize the fault tolerance, there exists coupling linkage among components and software modules that result in the structural interaction.
- Reconfiguration dependency: The fault tolerant strategy used in system may bring reconfiguration when failure occurs. For example, the primary sever can be switched on standby server when it fails in hot standby dual machine system.
- Maintenance dependency: If some failures occur, its maintenance capability is related to maintenance resource, maintenance man and repair time that always lead to maintenance dependency.

With these four kinds of fault dependencies in system, availability model of communication devices can not be built up easily. On the other hand, with the rapid development of computer technology and the emergence of new advanced communication applications, the higher reliability demand for the key field becomes more and more urgent. So fault tolerance techniques are widely applied in safe critical computer networks. Fault tolerance techniques are based on the idea of redundancy, namely, high availability of the system can be obtained by adding more backup devices. According to different position device located in network, redundancy techniques in network can be divided into web server fault tolerance techniques, techniques for linkage, and fault tolerance techniques on workstations. Since web server is a typical data-exchange device whose performance determines integrity and restorability of data in system directly, fault tolerance techniques for web server become the most important part in hardware fault tolerance techniques. Among large mount of web server fault tolerance techniques, cluster technique has been completely adopted with high reliability and retractility.

Although server cluster technique can improve system's availability, it makes the system and failing process more complex. As a typical system with fault dependencies, the failure in such system results from activations of hardware and/or software faults. Even though many publications have been devoted to the availability analysis of the combined hardware and software systems, work on both aspects dealt with at the same time is not prevalent. Moreover, even when hardware and software aspects are considered together for real systems, the sources of faults are not explicitly distinguished. Traditional fault tree analysis (FTA) can only describe logical combinations among the elements in system (e.g. processor, system disc and software, etc), but it is helpless to depict the fault redundancy among the different components. Furthermore, the continuous time Markov chain (CTMC) is difficult to analyze the fault-tolerant server system's availability for complexities in modeling and resolving. With advantage of explicitly and modularly modeling, Petri-net based models have been extensively used for performance modeling to analyze computer and communication systems. In reliability modeling community, Petri-net based models have received more and more attention. Compared with FTA and CTMC, Petri net is more suitable for modeling and easier to get results. M. Malhotra discussed the model of multiprocessor system

978-1-4244-1676-9/08 /$25.00 ©2008 IEEE

with generalized stochastic Petri nets (GSPN), and K. Kanoun built up the model of hardware and software component-interactions with GSPN. However, the integral availability model of fault-tolerant server system with consideration of fault dependencies has not been discussed before. Based on these reasons, this paper adopts GSPN technique to construct availability model of a typical fault-tolerant server system.

The rest of the paper is organized as follows. Section II analyses the structure and failure mode of the fault-tolerant server system. Section III describes availability models of systems with GSPN. Section IV presents the concluding remarks.

II. STRUCTURE AND FAILURE MODE OF SYSTEM

A typical fault-tolerant server system is double-machine and double-control system (see Figure.1). The system composes of server A and server B. Each server has its own system disc, which is used to set up system software, database software and application software. Share data discs connect with server A and server B by SCSI cables. In the course of operation, server A and server B run different applications, and data produced are stored in the share data discs. At the same time, two servers backup for each other. One server detects another's heart-beat by inner network or serial comport. When a server fails, another will take over the applications running on this faulted server to continue providing services for terminals.

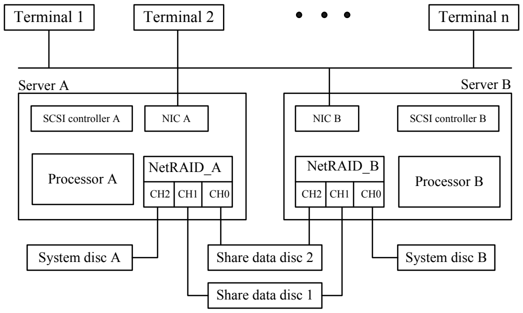

Figure 1. Structure of double-machine double-control system

Based on the system's structure and working procedure, the system's failure tree can be obtained as shown in Fig.2.

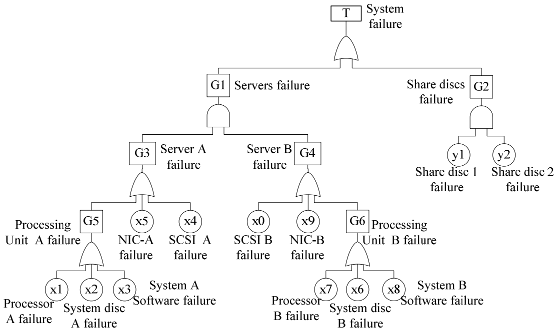

Figure 2. Failure tree of system

As be seen in Figure 2, FTA can only describe logical combinations among the elements of system (e.g. processor, system disc and software, etc) while it is difficult to deal with fault dependencies between hardware and software which exits in processing cell. Actually, software may fail due to errors taken place itself and interaction by failed processor and system disc. For example, the temporary faults in hardware such as processors superheated or system disc overflowing will lead the software out of order, and in extreme case, it will result system completely failing.

To simplify analysis, this paper only discusses faults interactions between hardware and software. Owing to the importance of the impact of temporary faults on the behavior of hardware and software, both permanent and temporary faults in hardware are considered. It is assumed that the activation of a hardware fault may lead to the following dependencies:

- An error due to the activation of a temporary hardware fault may propagate to the hosted software.
- An error due to the activation of a permanent fault in a computer leads to stop the hosted software that is restarted after the end of repair.

To describe faults interaction more accurately, this paper adopts GSPN technique to build up the system model.

A. *Availability model of the processing cell*

The processor, system disc and hosted software compose of the processing cell which is a typical element with fault dependency between hardware and software.

To simplify analysis, the models of hardware and software are built up respectively with GSPN. Considering the fault activation between hardware and software, integral availability model of the cell is set up subsequently. Figure 3 shows hardware GSPN availability model.

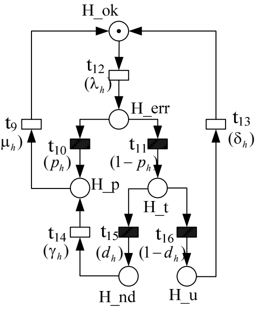

Figure 3. Hardware GSPN model of processing cell

The hardware in cell could fail due to many factors while it runs in a proper service state, then a token is transmitted to the place H_err from the place H_ok. The fault happened may be a permanent fault with the probability p_h, and the place H_p will produce a token. In this state, hardware needs repair wholly, and subsequently, hardware is repaired with the mean maintenance rate μ_h. Or the fault happened may be a temporary fault with the probability $1-p_h$, which may convert into permanent fault or be removed eventually by hardware itself with operating. A token is transmitted into the place H_nd with probability d_h, which will convert to H_p with the mean rate γ_h. A token is transmitted into the place H_u with probability $1-d_h$, and it will convert to H_ok with the mean rate δ_h.

Figure 5 shows hardware availability vs. λ_h and d_h.

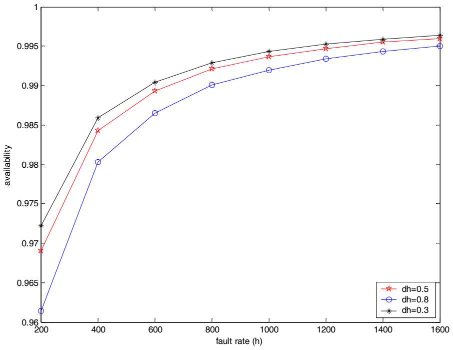

Figure 4. availability of hardware vs. failure rate

In Figure.4, probability d_h reflects effect of surrounding factors on hardware operation. The stronger the surrounding factors affect the system (viz. the larger the d_h is), the higher the failure probability is. In addition, parameter γ_h denotes the intensity of surrounding effects wherein the time from temporary fault to permanent failure becomes short with γ_h increasing.

Similarly, we can also establish the GSPN availability model of software shown in Figure 5.

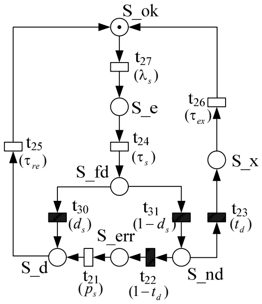

Figure 5. Software GSPN model of processing cell

Software on processing cell will fail with the rate λ_s by reason of many factors while it runs in the state of proper service, then a token is transmitted into the place S_e from the place S_ok. When an error happens, system will perform fault detection to the software with the mean rate τ_s. A perceived error with probability d_s may be found after system detection, so a token is transmitted into the place S_d and the software will finish restart with the mean rate τ_{re} subsequently. Undetected error (in this state, the place S_nd has a token) with probability $1-d_s$ may convert to perceived error with the mean rate p_s with the software running, and software will restart subsequently. Some undetected error with the probability t_d will be removed by system with the mean rate τ_{ex} with the software running, and software comes back to proper service state.

In Figure 6, probability t_d reflects effect of surrounding circumstance factors on software operation. The stronger the surrounding factors affect the system (viz. the smaller the probability t_d is), the higher the failure transmission probability that temporary fault converts to permanent failure occurs. In addition, parameter p_s denotes the intensity of surrounding effects wherein the time from temporary fault to permanent failure becomes short with p_s increasing.

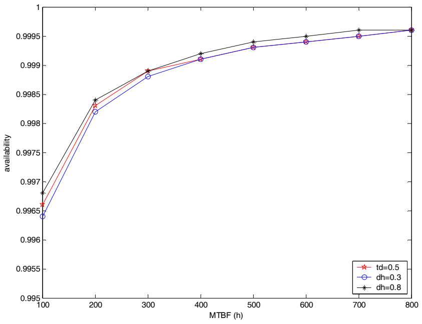

Figure 6. Software availability vs. λ_s and t_d

Combining the hardware and software availability models under consideration of fault dependencies, the integral GSPN availability model of the processing cell can be obtained as shown in Figure 7.

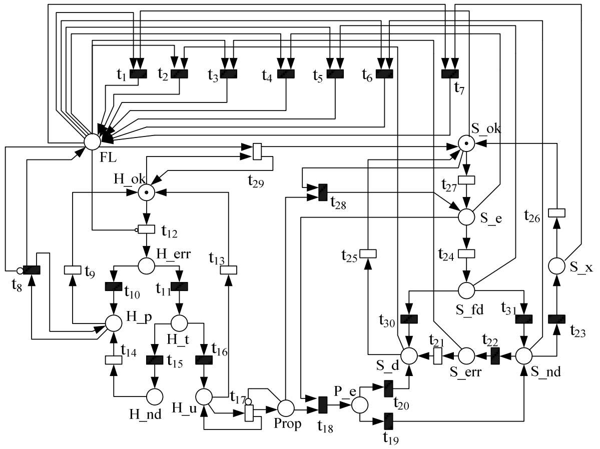

Figure 7. GSPN model of processing cell.

Herein hardware is in the state of H_u, it will recover with the mean rate δ_h. According to the assumptions in section II, temporary fault in hardware may propagate to software, namely, a token will transmitted into the place Prop and this will result in software's following behaviors:

- if a token is in place S_ok, it will turn to S_e.
- if a token is in place S_e, it will be transmitted to the place S_d with the probability d_s' or to the place S_nd with the probability $1-d_s'$.

On the other hand, an error due to the activation of a permanent fault in hardware leads to stop the hosted software that is restarted after the end of repair. Namely, while hardware is in the state of H_p, the cell will lose its function and a token is transmitted into FL, and the token in the software model will be eliminated by immediate transitions t_1~t_7. After hardware finishing repair and a token moving to the place H_ok, system will be restored at rate τ_{re}, and tokens will be transmitted into the place H_ok and S_ok. At the same time, the token in FL is removed.

Table I lists the states that occurs more frequently in course of GSPN model simulation.

TABLE I. EFFECTIVE STATES OF THE GSPN MODEL

state	H_ok	H_e	H_t	H_u	H_p	H_nd	prop	P_e
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0
6	1	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0	0

state	S_e	S_nd	S_fd	S_x	S_e	S_ok	S_d	FL
1	0	0	0	0	0	1	0	0
2	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	1	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	1

State 1 in Table I denotes that hardware and software in the cell are both in proper service state where the cell can fulfill its function well. State 2 presents that hardware is in good condition while software occurs an error. Because the error taken place in software may be translated into perceived fault and be removed subsequently, so sate 2 is a temporary state and it is not defined as the failure state. State 3 denotes that hardware is in proper service state while software is in permanent state and needs restoration which is one of the failure states in the cell. State 4 presents that software is in temporary fault state. Although in this state performance will be degraded, the cell can finish its basic function and performance will recover with the fault removed. Same to state 4, state 5 is the interim state where temporary fault in hardware may be translated into permanent failure or be removed. State 6 denotes that hardware has finished repair, and the cell will restart subsequently. State 7 denotes that hardware is out of order where system needs repair wholly.

Based on analysis above, it can be seen that the cell in state 3, 6 and 7 will lose it function completely and need restart or repair, so we define these three states as the failure state of the cell and the others as operational state.

To the combined hardware and software system, availability is not the simple product of reliabilities of hardware and software. It can be proved that the traditional availability of devices is some higher than actual value. And it is especially true when the hardware has higher fault rate (as shown in Figure 8).

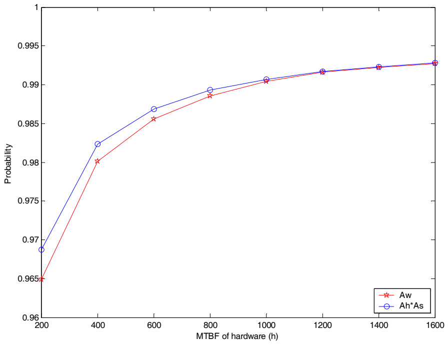

Figure 8. Availability of the cell

In order to simplify analysis on the processing cell, we suppose that the cell has only two states: failed and operational. Figure 11 describes the simplified GSPN availability model with equivalent mean failure rate λ_d and mean maintenance rate μ_d. Simplified GSPN model is a basic repairable system. D.up in Figure 9 denotes the operational state of the cell, and D.dn denotes the failed state of the cell. The most important thing about the simplified model is to determine equivalent failure rate λ_d and maintenance rate μ_d.

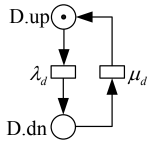

Figure 9. Simplified GSPN model of the cell

According to effective states of the cell in Table I, the failure states of processing cell compose of states 3, 6 and 7. The mean time that cell spends in the state 6 is $1/\tau_{re}$. The mean time that cell spends in the state 7 is $1/\mu_h$. And the mean time that cell spends in the state 3 is $1/\tau_{re}$. As a result, the total mean time that cell spends in failure state is:

$$T = \frac{\left[P(stat=3)+P(stat=6)\right]/\tau_{re}+P(stat=7)/\mu_h}{P(stat=3)+P(stat=6)+P(stat=7)}$$

Define that $\mu_d = 1/T$ is equivalent mean repair rate of the simplified model. According to repairable system theory [14], the equivalent failure rate λ_d is:

$$\lambda_d = \frac{(1-A_w)}{A_w}\mu_d$$

B. GSPN model of the redundant servers

Redundant server system consists of the server A and B. Each server includes a processing cell, a network interface card (i.e. NIC) and a SCSI controller. Form the view of reliability engineering, the server is a series system. Figure 7 depicts the GSPN availability model of the server B.

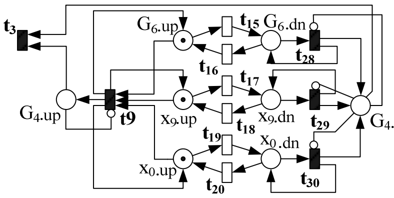

Figure 10. GSPN availability model of the server B.

The processing cell, the NIC and the SCSI controller will fail with the mean rate denoted by the immediate transitions t_{15}, t_{17} and t_{18} respectively. And they can finish maintenance with the mean rate denoted by the immediate transitions t_{16}, t_{18} and t_{20}. The server B is operational as long as the processing cell, the NIC and the SCSI controller are operational. In this state, the places G_6.up, x_9.up and x_0.up have a token respectively. The immediate transition t_9 fires, and the place G_4.up will produce a token. If any component among three fails, the place G_4.dn will produce a token and eliminate the token in G_4.up by the immediate transition t_3. In this state, the server B fails. Once the failed part accomplishes repair and tokens appear in places G_6.up, X_9.up and X_0.up simultaneously, G_4.up will produce a token and the token in G_4.dn will be cleaned. A failure-repair cycle finishes.

By simulation with the failure rates and maintenance rates given in Table IV, availability of the server B is 0.972547.

TABLE II. FAILURE RATES AND MAINTENANCE RATES IN SERVER

	cell	**NIC**	**SCSI Controller**
Failure rate ($\times 10^{-7}/s$)	3.47	3.47	2.77
Maintenance rate ($\times 10^{-5}/s$)	1.157	27.8	1.157

According to the system operating principle described in section II, the server A and B compose of a parallel system. And Figure.11 shows the GSPN availability model of the redundant servers system.

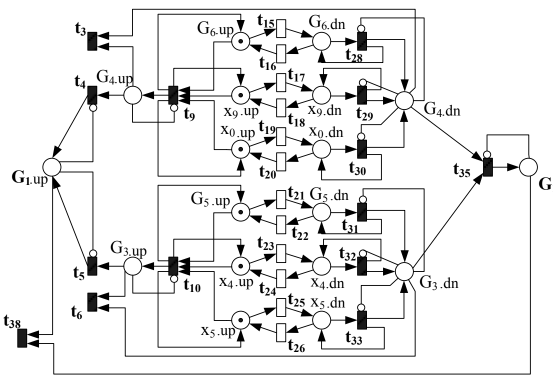

Figure 11. Availability model of redundant servers system.

While at least one token is in G_4.up and G_5.up, the place G_1.up has a token and the redundant servers system runs properly. Only if both the servers A and B fail, tokens will be transmitted into the place G_3.dn and G_4.dn. The place G_1.dn will produce a token after t_{35} firing and the token in G_1.up will be eliminated by t_{38}. In this state, the system is out of operation.

C. GSPN model of redundant discs system

Figure.12 shows the GSPN availability model of the redundant data discs system.

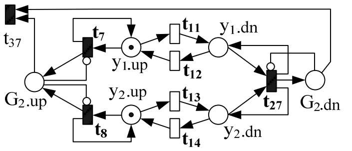

Figure 12. GSPN availability model of redundant discs system.

Redundant data discs system composes of data disc 1 and 2. During operation, the disc 1 and 2 may fail at the mean rate denoted by timed transitions t11 and t_{12} respectively. Discs will recover from the failure state at the mean maintenance rates denoted by t_{13} and t_{14}. If at least one disc works properly, G_2.up will have a token and the system runs in the proper service state. If the discs 1 and 2 fail simultaneously, y_1.dn and y_2.dn both produce tokens and a token is transmitted into G_2.dn after t_{27} firing. The token in G_2.up will be eliminated by t_{37}. In this state, the redundant discs system fails. The system will recover from the failure state after any disc finishing maintenance.

D. GSPN model of the system

The integral GSPN availability model of the fault-tolerant server system can be derived from the availability models of the redundant servers system and the redundant discs system. Figure 10 shows the integral availability model.

Form the view of reliability engineering, redundant servers and discs compose of a series system. When the redundant servers and discs are both in proper service state, G_2.up and G_1.up have tokens. T.up will produce a token after t_2 firing. In this state, the system works properly. If any subsystem fails, G_2.dn or G_1.dn will produce a token. A token will transmitted into T.dn and the token in T.up will be eliminated. In this state, system fails and needs maintenance.

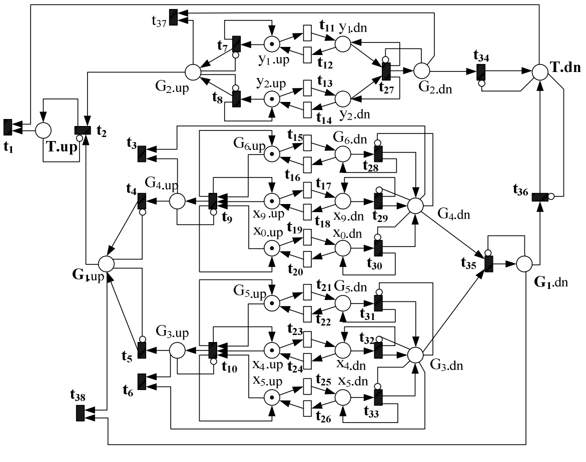

Figure 13. GSPN availability model of the system.

By simulation, we can get the system's effective states which appear more frequently than others shown in Table.III.

TABLE III. THE EFFECTIVE STATES OF THE SYSTEM

state	y1.up	y1.dn	y2.up	y2.dn	G2.dn	G2.up	G6.up	G6.dn	x9.up	x9.dn	x0.up	x0.dn	G4.dn
1	1	0	1	0	0	1	1	0	1	0	1	0	0
2	1	0	1	0	0	1	0	1	1	0	1	0	1
3	1	0	1	0	0	1	1	0	1	0	1	0	0
4	1	0	1	0	0	0	0	1	1	0	1	0	1

state	G4.up	G5.up	G5.dn	x4.up	x4.dn	x5.up	x5.dn	G3.dn	G3.up	G1.up	T.up	G1.dn	T.dn
1	1	1	0	1	0	1	0	0	1	1	1	0	0
2	0	1	0	1	0	1	0	0	1	1	1	0	0
3	1	0	1	1	0	1	0	1	0	1	1	0	0
4	0	0	1	1	0	1	0	1	0	0	0	1	1

Assume that the failure rate of the disc is 2.77×10^{-7} /s and repair rate is 1.157×10^{-5} /s, and the failure rate and repair rate are taken from Table IV. The steady probabilities of states in system model are listed in table IV.

TABLE IV. THE STEADY PROBABILITIES OF THE SYSTEM'S STATES

state	1	2	3	4
Pr.	0.940341	0.021387	0.024362	0.000164

Availability of the fault-tolerant server system is 0.999836. Compared with the single server system shown in Figure.10, availability enhances greatly by adopting redundancy technique.

III. CONCLUSIONS

Since web server is a typical system with fault dependencies whose performance determines integrity and restorability of data in system directly, fault tolerance techniques for web server become the most important part. In order to improve the server availability, cluster technique is widely used. Although using server cluster technique can improve system's availability, it makes system more complex and makes availability analysis more difficult.

This paper sets up availability model of the processing cell, redundant servers sub-system and availability model of redundant data discs by GSPN which has an advantage of modeling explicitly and modularly. Combining these models, we can achieve availability model of the system. By analysis, we can draw the conclusions as flows:

(1) The system's availability enhances greatly by adopting redundancy technique.

(2) The server is a typical fault dependency system. On the one hand, faults of processor and system disc can make software failed. Failures of software can be eliminated by restarting system, and failures of hardware can be recovered by replacing fault devices. It is shown that software failure is derived form both software errors and hardware failure. With the failure rate of hardware decreasing, software failure only depends on software errors primarily.

According to the availability analysis for the fault-tolerant server system, we can find the system's weakness and get the high availability by selecting reasonable failure rates and repair rates of the components in the system. The research method adopted in this paper can be used widely in redundant system with integrated hardware and software.

ACKNOWLEDGMENT

The research was result of a project financed by national fund. Thanks the support of National Natural Science Foundation (60371043), National Defense Foundation of China (51419020404HK0150), Program 111, Program for New Century Excellent Talents in University （NCET） and Beijing Teaching Innovation Program.

REFERENCES

[1] Karama Kanoun, "Fault-Tolerant System Dependability – Explicit Modeling of Hardware and Software Component-interactions", IEEE Trans. Rel., vol.49, pp. 363-375, 2000.

[2] Karama Kanoun, Marie Borrel. "Availability of CAUTRA, a Subset of the French Air Traffic Control System", IEEE Transaction on Reliability Vol.48, pp. 528-535, 1999.

[3] Singpurwalla ND, Kong CW. Specifying interdependence in networked systems. IEEE TRANSACTIONS ON RELIABILITY 2004, 53: 401-405.

[4] Wang Shaoping. Engineering Reliability. Beijing University of aeronautics and Astronautics Press, 2000.

[5] Liu Weiming, "Fault Tolerant Analysis for Computer Networks", Microcomputer & Its Applications, vol.8, pp. 34-35.1992

[6] Dai YS, Xie M, Poh KL. Modeling and analysis of correlated software failures of multiple types. IEEE TRANSACTIONS ON RELIABILITY 2005, 54: 100-106.

[7] Manish Malhostra, Kishor S. Trivedi, "Dependability Modeling Using Petri-Nets", IEEE Trans. Rel. vol. 44, pp. 428-440, 1995.

[8] Winfrid G. Schneeweiss, "Tutorial: Petri Nets as a Graphical Description Medium for Many Reliability Scenarios", IEEE Transactions on Reliability, Vol.50, pp:159-164, 2001

[9] Ming-Lei Yin, Craig L. Hyde, Larry E. James, "A Petri-Net Approach for Earl-Stage System-Level Software Reliability Estimation", IEEE 2000 Proceedings Annual Reliability and Maintainability Symposium, pp:100-105, 2000.

[10] Lianzhang Zhu, Yanchen Li, "Reliability Analysis of Component Software based on Stochastic Petri Nets", International Conference on Computer and Information Science, 2007.

[11] Vitali Volovoi, "Modeling Multiphased Missions Using Stochastic Petri Nets with Aging Tokens", RAMS2004, pp:232-236, 2004.