Multi-State Availability Modeling in Practice

Kishor S. Trivedi, Dong Seong Kim, Xiaoyan Yin

Depart ment of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA
kst@ee.duke.edu, {dk76, xy15}@duke.edu
http://dhaal.ee.duke.edu

Abstract. This chapter presents multi-state availability modeling in practice. We use three analytic modeling techniques; (i) continuous time Markov chains, (ii) stochastic reward nets and (iii) multi-state fault trees. Two case studies are presented to show the usage of these modeling techniques: a simple system with two boards and the processors subsystem of the VAXcluster. The three modeling techniques are compared in terms of the solution accuracy and the solution time.

1 Introduction

There are systems which have multiple states and whose components also have multiple states. In order to capture such multi-state system availability, many techniques have been proposed. In this chapter, we provide three analytic modeling techniques which are useful in modeling multi-state availability; continuous time Markov chains (CTMC), stochastic reward nets (SRN) [3] and multi-state fault trees (MFTs). Two case studies are shown including a system with two boards and the processors subsystem of the VAXcluster.

The rest of this chapter is organized as follows. Multi-state reliability modeling techniques for a simple system with two boards are described in Section 2. Multi-state availability modeling techniques for the processors subsystem of the VAXcluster are described in Section 3. The three modeling techniques are compared in terms of model solution accuracy and execution time. The chapter concludes in Section 4.

2 Two boards system

Figure 1 shows a system with two boards (B_1 and B_2) where each board has a processor and a memory [5]. The memories (M_1 and M_2) are shared by both the processors (P_1 and P_2). The processor and memory on the same board can fail separately, but statistically-dependently. Assume that the time to failure of a processor and the time to failure of a memory are exponentially distributed with rates λ_p and λ_m, respectively. The time to the common cause failure (i.e., both a processor and the memory on the same board fail simultaneously) is exponentially distributed with rate λ_{mp}.

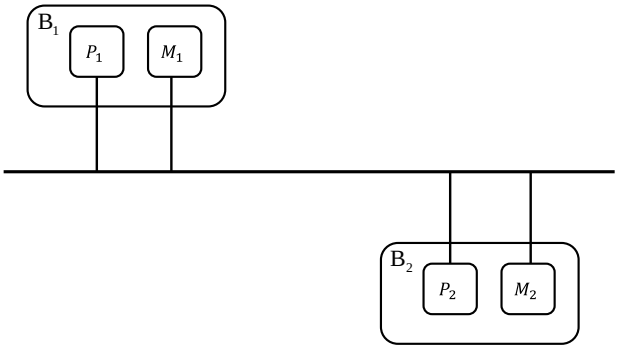


Fig. 1. The two boards system

We consider each board as a component with four states:

- Component State 1: both *P* and *M* are down.
- Component State 2: *P* is functional, but *M* is down.
- Component State 3: *M* is functional but *P* is down.
- Component State 4: both *P* and *M* are operational.

The system states are defined as follows:

- System State 1 (S_1): either no processor or no memory is operational and hence the system is down.
- System State 2 (S_2): at least one processor and exactly one memory are operational.
- System State 3 (S_3): at least one processor and both of the memories are operational.

2.1 Continuous time Markov chain model

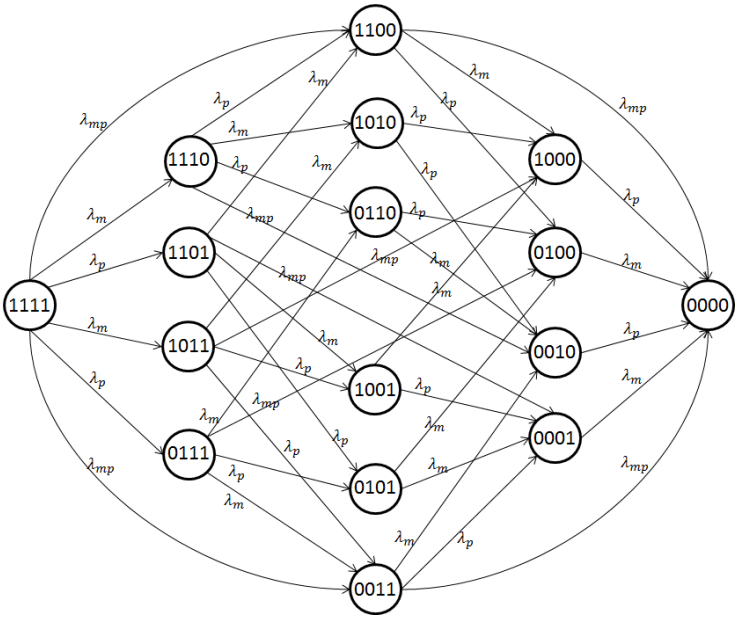


Fig. 2. CTMC model for the two boards system

A complete (homogeneous) continuous time Markov chain (CTMC) reliability model is constructed to capture the system's behavior as shown in Figure 2. The states of the Markov chain are represented by ($P_1M_1P_2M_2$), where *1* denotes up and *0* denotes down for each device. Denote the probability that the system in state *i* at time *t* by $\pi_{Si}(t)$. By solving the CTMC model and using reward rate assignment, we compute the expected reward rate at time *t,* $\pi_{Si}(t)$ using the following formula:

$$\pi_{S_i}(t) = \sum_j r_{i,j} \cdot \pi_j(t) \tag{1}$$

where the reward rate assignment is:

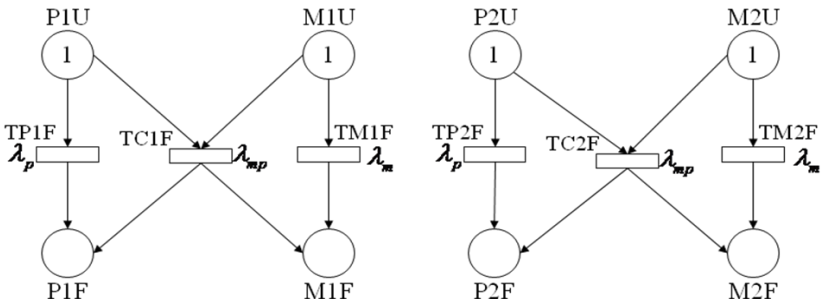
$$\begin{aligned}
r_{1,j} &= \begin{cases} 1 & j = (1010),(0101),(1000),(0100),(0010),(0001),(0000) \\ 0 & otherwise \end{cases} \\
r_{2,j} &= \begin{cases} 1 & j = (1110),(1011),(1100),(0110),(1001),(0011) \\ 0 & otherwise \end{cases} \\
r_{3,j} &= \begin{cases} 1 & j = (1111),(1101),(0111) \\ 0 & otherwise \end{cases}
\end{aligned} \tag{2}$$

Table 1 shows the expected reward rate at time t, $\pi_{Si}(t)$ of the CTMC model at different time epochs with parameters, $1/\lambda_p=1000$ hours, $1/\lambda_m=2000$ hours and $1/\lambda_{mp}=3000$ hours.

Table 1. Results of the CTMC model for the two boards system

t (hour)	$\pi_{S_1}(t)$	$\pi_{S_2}(t)$	$\pi_{S_3}(t)$
0	0	0	1
100	2.03280585e-002	1.38357902e-001	8.41314039e-001
200	7.02653699e-002	2.29253653e-001	7.00480977e-001
300	1.37122363e-001	2.84435223e-001	5.78442414e-001
∞	1	0	0

2.2 Stochastic reward net model



(a) (b)

Fig. 3. SRN model for the two boards system

Figure 3 shows the stochastic reward net (SRN) model for the two boards system. Figure 3(a) represents the failure behavior of the processor P_1 and the memory M_1. If a token is in place *P1U*, the processor is operational, otherwise the processor is down. Figure 3(a) also presents the failure behavior of the memory M_1; if one token is in place *M1U*, the memory is operational, otherwise the memory is down. The transition *TC1F* represents a common-cause failure. The transition *TC1F* is enabled when there is one token in each place *P1U* and place *M1U*. Similarly, Figure 3(b) shows the failure behavior of the processor P_2 and the memory M_2. We can compute the probability that the system is in state i, denoted as $\pi_{Si}(t)$, using the definition of reward functions in SHARPE [4] as shown in Equation (3).

$$r_{S1}=\begin{cases}1 & \left(\#\left(\mathrm{P1U}\right)+\#\left(\mathrm{P2U}\right)==0\right)\mathrm{or}\left(\#\left(\mathrm{M1U}\right)+\#\left(\mathrm{M2U}\right)==0\right)\\ 0 & otherwise\end{cases}$$

$$r_{S2}=\begin{cases}1 & \left(\begin{array}{l}\left(\left(\#\left(\mathrm{P1U}\right)+\#\left(\mathrm{P2U}\right)>=1\right)\mathrm{and}\left(\#\left(\mathrm{M1U}\right)==1\right)\mathrm{and}\left(\#\left(\mathrm{M2U}\right)==0\right)\right)\\ \mathrm{or}\left(\left(\#\left(\mathrm{P1U}\right)+\#\left(\mathrm{P2U}\right)>=1\right)\mathrm{and}\left(\#\left(\mathrm{M2U}\right)==1\right)\mathrm{and}\left(\#\left(\mathrm{M1U}\right)==0\right)\right)\end{array}\right)\\ 0 & otherwise\end{cases} \quad (3)$$

$$r_{S3}=\begin{cases}1 & \left(\#\left(\mathrm{P1U}\right)+\#\left(\mathrm{P2U}\right)>=1\right)\mathrm{and}\left(\#\left(\mathrm{M1U}\right)+\#\left(\mathrm{M2U}\right)==2\right)\\ 0 & otherwise\end{cases}$$

Table 2 shows the results of $\pi_{Si}(t)$ at different times using the SRN model.

Table 2. Results of the SRN model for the two boards system

t (hour)	$\pi_{S_1}(t)$	$\pi_{S_2}(t)$	$\pi_{S_3}(t)$
0	0	0	1
100	2.03280585e-002	1.38357902e-001	8.41314039e-001
200	7.02653699e-002	2.29253653e-001	7.00480977e-001
300	1.37122363e-001	2.84435223e-001	5.78442414e-001
∞	1	0	0

Comparing the values in Table 1 and Table 2, we can see that the CTMC model and the SRN model have identical results.

2.3 Multi-state fault trees model

An alternative approach to obtain the probability of the system states is to use multi-state fault trees (MFTs). By the definition in the beginning of Section 2, each board

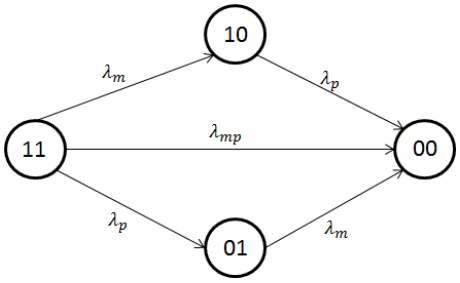


Fig. 4. CTMC model for a single board

is considered as a component with four states. The failure behavior of a single component can be captured by the CTMC model in Figure 4. The states of the Markov chain is represented by (*PM*), where *1* denotes up and *0* denotes down for each device.

By solving the CTMC model in Figure 4, the transient probabilities for different component states $\pi_{Bi,j}(t)$, where $B_{i,j}$ denotes the board B_i being in state j, can be obtained. Equation 4 shows the formulas for $\pi_{Bi,j}(t)$.

$$\pi_{B_{i,j}}(t)=\begin{cases}\pi_{00}(t)=\dfrac{\lambda_m\lambda_p-\lambda_{mp}{}^2}{(\lambda_m+\lambda_{mp})(\lambda_p+\lambda_{mp})}e^{-(\lambda_m+\lambda_p+\lambda_{mp})t}-\dfrac{\lambda_p}{\lambda_p+\lambda_{mp}}e^{-\lambda_m t}-\dfrac{\lambda_m}{\lambda_m+\lambda_{mp}}e^{-\lambda_p t}+1, & j=1\\ \pi_{10}(t)=\dfrac{\lambda_m}{\lambda_m+\lambda_{mp}}\left[e^{-\lambda_p t}-e^{-(\lambda_m+\lambda_p+\lambda_{mp})t}\right], & j=2\\ \pi_{01}(t)=\dfrac{\lambda_p}{\lambda_p+\lambda_{mp}}\left[e^{-\lambda_m t}-e^{-(\lambda_m+\lambda_p+\lambda_{mp})t}\right], & j=3\\ \pi_{11}(t)=e^{-(\lambda_m+\lambda_p+\lambda_{mp})t}, & j=4\end{cases} \tag{4}$$

Based on the states for each component (i.e., a board), the two boards system states can be obtained using MFTs model as shown in Figure 5.The definition of the system states is presented in the beginning of Section 2.

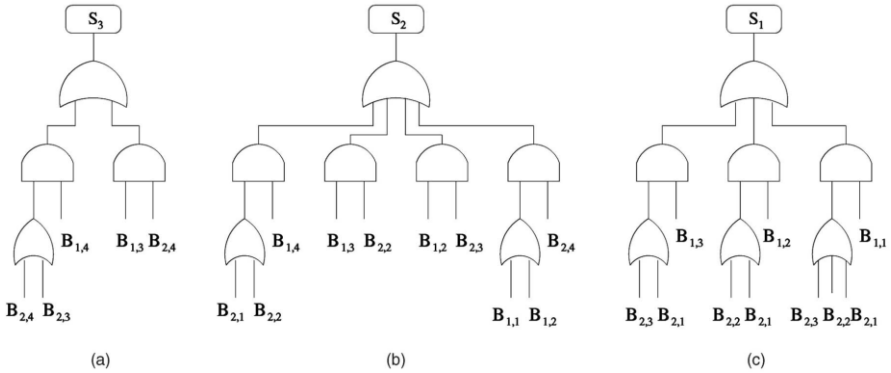


Fig. 5. MFTs model for the two boards system

Table 3. Results of the MFTs model for the two boards system

t (hour)	$\pi_{S_1}(t)$	$\pi_{S_2}(t)$	$\pi_{S_3}(t)$
0	0	0	1
100	2.03223700e-002	1.38269880e-001	8.41407750e-001
200	7.02405700e-002	2.29213640e-001	7.00345800e-001
300	1.37092520e-001	2.84365240e-001	5.78342250e-001
∞	1	0	0

For any given time t, we can obtain the value of $\pi_{Bi,j}(t)$ by solving the CTMC model in Figure 4 and Equation 4, which is then assigned as the probability for event $B_{i,j}$ in

the MFTs model in Figure 5. By solving the MFT model, the probability for the system in state i, denoted as $\pi_{Si}(t)$, can be obtained. Table 3 shows the results for the different value of t, given that $1/\lambda_p$=1000 hours, $1/\lambda_m$=2000 hours and $1/\lambda_{mp}$=3000 hours. By comparing the values in Table 1 with them in Table 3, the multi-state fault tree model and the CTMC model (in section 2.1) show nearly the same results. Differences are due to numerical solution errors.

2.4 Model Comparison

Note that from section 2.1, 2.2 and 2.3, similar results are obtained using three different analytic modeling techniques including CTMCs, SRNs, and MFTs. However, the states space of the CTMC model will increase exponentially as the number of boards increase. If the number of boards is n, the number of states will be 2^{2n}. If n becomes large, this approach will face a state-space explosion problem. SRN model is easy to construct among these three model types, but the underlying Markov chain for the SRN model also faces state explosion problem when n becomes very large. By contrast, the multi-state fault tree model still consists of three fault trees when n increases, which is more efficient to solve than the CTMC model and the SRN model. However, the construction of such MFT model will consume more effort and sometimes error-prone than the CTMC model and the SRN model.

3 VAXcluster system

A VAXcluster is a closely-coupled multicomputer system that consists of two or more VAX processors, one or more storage controllers (HSCs), a set of disks and a star coupler (SC). The star coupler can be omitted from the model since it is extremely reliable. In this Section, we concentrate on the analysis using different models for the processing sub-system in the VAXcluster [1][2].

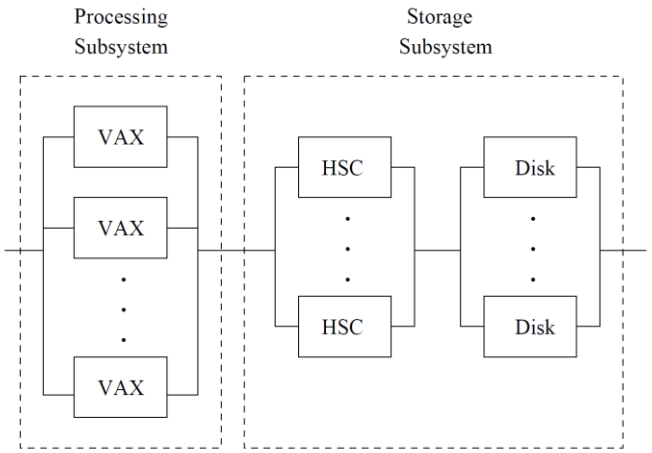


Fig. 6. A VAXcluster system

3.1 Continuous time Markov chain model

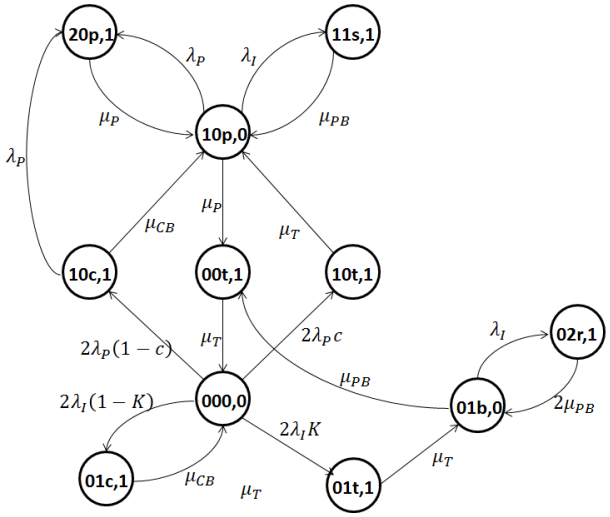


Fig. 7. CTMC model for the two-processor VAXcluster

A CTMC availability model can be developed for the processing sub-system in the VAXcluster [1][2] in which two types of failure and a coverage factor for each failure type exist. By using a single state space model, shared repair for the processors in a cluster is taken into account. The times to failures, the repair times and other recovery times are all assumed to be exponentially distributed. A CTMC availability model for

a two-processor VAXcluster is shown in Figure 7. A processor can suffer two types of failures: permanent and intermittent. A processor recovers from a permanent failure by a physical repair and from an intermittent failure by a processor reboot. These failures are further classified into covered and not covered. A covered processor failure causes a brief (in the order of seconds) cluster outage to reconfigure the failed processor out of the cluster and back into the cluster after it is fixed. Therefore, a covered failure causes a small loss in system up time. A not-covered failure causes the entire cluster to go down until it is rebooted. The parameters for the models are chosen as follows:

- Mean time to permanent failure: $1/\lambda_P = 5000$ hours
- Mean time to intermittent failure: $1/\lambda_I = 2000$ hours
- Mean processor repair time: $1/\mu_P = 2$ hours
- Mean processor reboot time: $1/\mu_{PB} = 6$ minutes
- Mean cluster reboot time: $1/\mu_{CB} = 10$ minutes
- Mean cluster reconfiguration time: $1/\mu_T = 30$ seconds
- Coverage factor for permanent failure: $c = 0.9$
- Coverage factor for intermittent failure: $K = 0.9$

The states of the Markov chain are represented by (abc,d), where,

$$a = number\ \ of\ \ processors\ \ down\ \ with\ \ permanent\ \ failure$$

$$b = number\ \ of\ \ processors\ \ down\ \ with\ \ \text{int}ermittent\ \ failure$$

$$c = \begin{cases} 0 & if\ \ both\ \ processors\ \ are\ \ up \\ p & if\ \ one\ \ processor\ \ is\ \ being\ \ repaired \\ b & if\ \ one\ \ processor\ \ is\ \ being\ \ rebooted \\ c & if\ \ cluster\ \ is\ \ undergoing\ \ a\ \ reboot \\ t & if\ \ cluster\ \ is\ \ undergoing\ \ a\ \ reconfiguration \\ r & if\ \ two\ \ processors\ \ are\ \ being\ \ rebooted \\ s & if\ \ one\ \ is\ \ being\ \ rebooted\ \ and\ \ the\ \ other\ \ is\ \ being\ \ repaired \end{cases}$$

$$d = \begin{cases} 0 & cluster\ \ up\ \ state \\ 1 & cluster\ \ down\ \ state \end{cases}$$

The system states are defined as

- System State 1 (S_1): System is up
- System State 2 (S_2): System is down due to cluster reboot
- System State 3 (S_3): System is down due to cluster reconfiguration
- System State 4 (S_4): System is down due to all processors are down (either under reboot or repair)

Denote the steady-state probability that the system is in each state as π_{Si}, where *i=1, 2, 3, 4*. We can solve the CTMC model to compute the steady-state probability for each state π_j and then use reward rate assignment to compute π_{Si} based on the following formula:

$$\pi_{S_i} = \sum_j r_{i,j} \cdot \pi_j \tag{5}$$

where,

$$\begin{aligned}
r_{1,j} &= \begin{cases} 1 & j = (000,0),(10p,0),(01b0) \\ 0 & otherwise \end{cases} \\
r_{2,j} &= \begin{cases} 1 & j = (10c,1),(01c,1) \\ 0 & otherwise \end{cases} \\
r_{3,j} &= \begin{cases} 1 & j = (10t,1),(00t,1),(01t,1) \\ 0 & otherwise \end{cases} \\
r_{4,j} &= \begin{cases} 1 & j = (11s,1),(20p,1),(02r,1) \\ 0 & otherwise \end{cases}
\end{aligned} \tag{6}$$

We computed the following results by solving the CTMC model using software package SHARPE [4] as shown in Table 4.

Table 4. Results of the CTMC model for the two-processor VAXcluster

	π_{S_1}	π_{S_2}	π_{S_3}	π_{S_4}
Steady-state probability	9.99955011e-001	2.33113143e-005	2.13134041e-005	3.64575688e-007

The main problem with this approach is that the size of the CTMC model grows exponentially with the number of processors in the VAXcluster system. The largeness posed the following challenges; (1) the difficulty in generating the state space and (2) the capability of the software to solve the model with thousands of states for VAXcluster system with *n>5* processors. Using SRN model instead of CTMC can avoid the difficulty in generating the state space. However, the underlying Markov chains generated by the SRN model still face the largeness problem when *n* becomes large and hence exceed the capability of the software to solve the model. The MFT model can avoid such difficulties by utilizing the features of combinatorial models. The following two sections present the SRN model and MFTs model respectively.

3.2 Stochastic reward net model

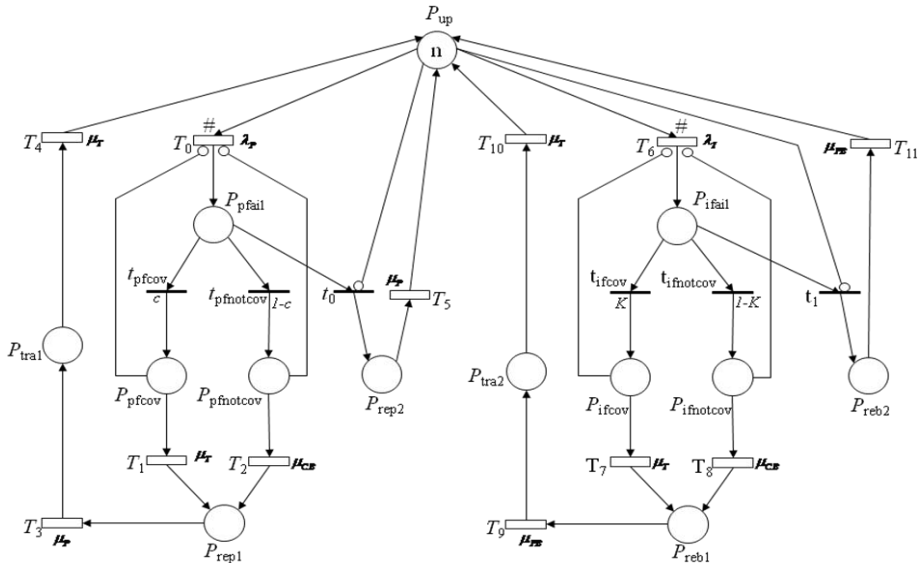


Fig. 8. SRN model for *n*-processor VAXcluster (using immediate transitions)

Figure 8 shows the SRN availability model for *n*-processor VAXcluster system. The number of tokens in place *Pup* represents the number of non-failed processors. The initial number of tokens in this place is *n*. As mentioned in Section 3.1, processors suffer two types of failure; permanent and intermittent and the failures can be covered or not. The firing of transition T_0 represents the permanent failure of one of the processors. The inhibitor arcs from the place P_{pfcov} and place $P_{pfnotcov}$ ensure that when the VAXclsuter system is undergoing a reconfiguration or reboot, no further failures occur. The firing rate of the transition T_0 is marking-dependent: Rate $(T_0)=\lambda_P \#P_{up}$, where λ_P is the permanent failure rate and $\#P_{up}$ is the number of tokens in place P_{up}. When a token appears in place P_{fail}, the immediate transitions t_{pfcov}, $t_{pfnotcov}$, and t_0 are enabled. If no token is in place P_{up}, then the immediate transition t_0 will be enabled and will be assigned a higher priority than t_{pfcov} and $t_{pfnotcov}$; this is done to ensure that for the last processor to fail and there then is no cluster reconfiguration or reboot delay. A token will be deposited in place P_{rep2} by firing immediate transition t_0. Otherwise t_{pfcov} or $t_{pfnotcov}$ will fire with probabilities c and $1\text{-}c$, respectively. In the case that t_{pfcov} (covered case) fires, cluster reconfiguration (T_1) takes place to remove the failed processor from the cluster with rate μ_T and then the failed processor is repaired (T_3) with rate μ_P and another cluster reconfiguration takes place to readmitting the repaired processor into the cluster with rate μ_T. In the case that $t_{pfnotcov}$ (not covered case) fires, the cluster is rebooted with rate μ_{CB}, the processor repair and cluster reconfiguration are followed.

The firing of transition T_6 represents intermittent failure of one of the processors. The firing rate of transition T_6 is marking-dependent: Rate $(T_6)= \lambda_I \#P_{up}$, where λ_I is the

failure rate. Similar to the permanent failure, if no token is in place P_{up}, then the immediate transition t_1 will be enabled and will be assigned a higher priority than t_{ifcov} and $t_{ifnotcov}$; this is done to ensure that for the last processor to fail, there is no cluster reconfiguration or reboot delay but only process reboot of the last processor. A token will be deposited in place P_{reb2} by firing immediate transition t_1 and the processor is rebooted with rate μ_{PB}, otherwise t_{ifcov} or $t_{ifnotcov}$ will fire with probabilities K and $1\text{-}K$, respectively. In the case that t_{ifcov} (covered case) fires, cluster reconfiguration (T_7) takes place to remove the failed processor from the cluster with rate μ_T and then the failed processor is rebooted (T_9) with rate μ_{PB} and another cluster reconfiguration takes place to readmitting the rebooted processor into the cluster with rate μ_T. In the case that $t_{ifnotcov}$ (not covered case) fires, the cluster is rebooted with rate μ_{CB}, the processor reboot and another cluster reconfiguration are followed.

We define reward rate functions for the system states in SHARPE [4] as follows according to the system states defined in section 3.1.

$$
\begin{aligned}
r_{S1} &= \begin{cases} 1 & \left(\begin{array}{l}(\#(\text{Pup})>=1)\text{and}(\#(\text{Ppfcov})+\#(\text{Ppfnotcov})+\\ \#(\text{P\_ifcov})+\#(\text{P\_ifnotcov})+\#(\text{Ptra1})+\#(\text{Ptra2})==0)\end{array}\right) \\ 0 & \textit{otherwise} \end{cases} \\
r_{S2} &= \begin{cases} 1 & \left((\#(\text{Ppfnotcov})>=1)\text{or}(\#(\text{P\_ifnotcov})>=1)\right) \\ 0 & \textit{otherwise} \end{cases} \\
r_{S3} &= \begin{cases} 1 & \left((\#(\text{Ppfcov})>=1)\text{or}(\#(\text{P\_ifcov})>=1)\text{or}(\#(\text{Ptra1})>=1)\text{or}(\#(\text{Ptra2})>=1)\right) \\ 0 & \textit{otherwise} \end{cases} \\
r_{S4} &= \begin{cases} 1 & \left((\#(\text{Prep1}))+(\#(\text{Prep2}))+(\#(\text{Preb1}))+(\#(\text{Preb2}))==n\right) \\ 0 & \textit{otherwise} \end{cases}
\end{aligned}
\tag{7}
$$

The results of the SRN model for two-processor VAX cluster are shown in Table 5. Compared to the results for the CTMC model in Table 4, we can see that SRN model showed nearly similar results of π_{Si}. We can easily compute the system availability for the n-processor VAXcluster using the SRN model. However, as shown in Section 3.4, the execution time of the SRN model exponentially increases when n becomes large, which makes the SRN model less efficient than the MFT model presented in the next section.

Table 5. Results of the SRN model for the two-processor VAXcluster

	π_{S_1}	π_{S_2}	π_{S_3}	π_{S_4}
Steady-state probability	9.99955087e-001	2.33168407e-005	2.13182409e-005	2.78074293e-007

3.3 Multi-state fault trees model

As mentioned in Section 3.1 and Section 3.2, the state-space of the CTMC model and underlying Markov chains of the SRN model for *n*-processor VAXcluster increases exponentially with *n*, thereby making the availability difficult to analyze at large values of *n*. In this section, an approximate analysis for an *n*-processor VAXcluster is developed to avoid the largeness associated with state space models. .

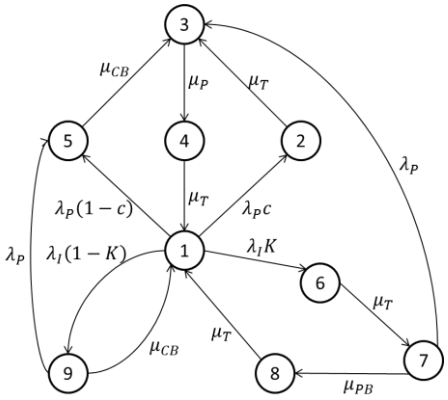


Fig. 9. CTMC model of a single processor

To obtain approximate system availability, the following assumptions are used:

1. The behavior of each processor is modeled by a homogeneous CTMC and assume that this processor did not break the quorum rule (i.e., at least one other processor is operational). This assumption is justified by the fact that the probability of VAXcluster failure due to loss of quorum (i.e., all processors are down) is relatively low.
2. Each processor has an independent repairman. This assumption is justified if the MTTF (mean time to failure) is large compared to the MTTR (mean time to repair) so that the time a faulty processor spends waiting for the repair crew to arrive is negligible.

These assumptions allow the decomposition of the *n*-processor VAXcluster into *n* independent subsystems, where each subsystem represents the behavior of one processor and can be modeled using the CTMC model shown in Figure 9. Furthermore, the states of such CTMC sub-model for the individual processors are classified into the following three super-states:

- Super-state 1: the set of states in which the processor is up = {1}.
- Super-state 2: the set of states in which the processor is undergoing reboot or repair and hence the processor is down = {3, 7}.

- Super-state 3: the set of states in which the cluster is undergoing reboot = {5, 9}.
- Super-state 4: the set of states in which the cluster is undergoing reconfiguration = {2, 4, 6, 8}

Therefore, by solving the CTMC sub-model and utilizing reward rate assignment, the steady-state probability for each super-state of a processor can be obtained. Subsequently, these super-states are considered as different states of a multi-state component (i.e., a processor) and multi-state fault trees can be constructed to compute the steady-state probability that the system is in each state, which is denoted as π_{Si} and defined in Section 3.1. The MFTs model is illustrated in Figure 10, where $P_{i,j}$ denotes that processor i is in super-state j.

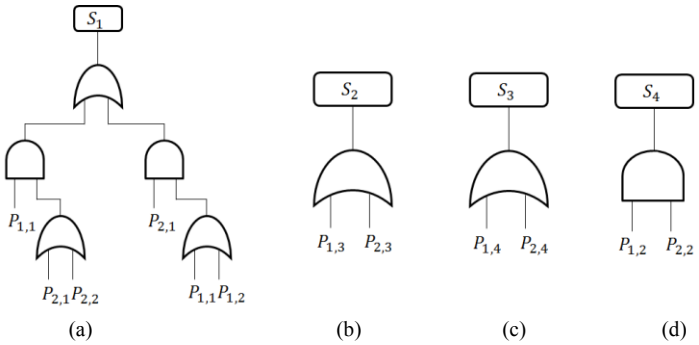


Fig. 10. MFTs model for the two-processor VAXcluster

The results for the MFTs model are shown in Table 6. Compared to the results for the complete CTMC model in Table 4, we can see that the MFTs model has good approximation for the exact results. Comparing to the difficulty in generating state spaces for a single complete CTMC model, we can easily extend the MFTs model to n-processor VAXcluster when n becomes large.

Table 6. Results of the MFTs model for the two-processor VAXcluster

	π_{S_1}	π_{S_2}	π_{S_3}	π_{S_4}
Steady-state probability	9.99955157e-001	2.33222980e-005	2.13232545e-005	1.97840042e-007

Based on the steady-state probabilities for different system states, the downtimes in hours per year can be computed and are shown in Figure 11. Here the downtime $D(n)=U(n)\times 8760$ hour per year and is expressed as Equation 8.

$$D(n)=U(n)\times 8760=\begin{cases}\left[1-\pi_{S_1}(n)\right]\times 8760 & Total\ \ downtime\\ \pi_{S_2}(n)\times 8760 & Downtime\ \ due\ \ to\ \ cluster\ \ reboot\\ \pi_{S_3}(n)\times 8760 & Downtime\ \ due\ \ to\ \ cluster\ \ reconfiguration\\ \pi_{S_4}(n)\times 8760 & Downtime\ \ due\ \ to\ \ all\ \ processors\ \ down\end{cases} \tag{8}$$

As shown in the results, the total downtime is not monotonically decreasing with the number of processors. The reason for this behavior is that as the number of processors increases beyond 2, the primary cause of downtime is the cluster reboot and reconfiguration and the number of reboots and reconfigurations nearly linearly increasing with the number of processors.

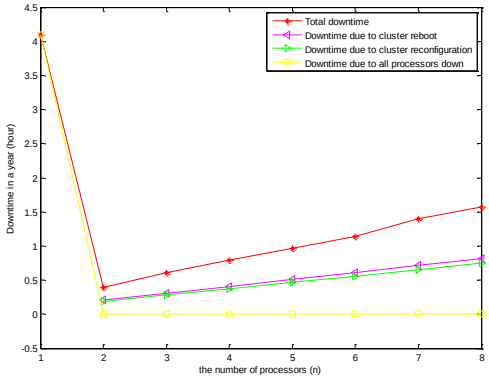


Fig. 11. Downtime Vs. the number of processors for the MFTs model

3.4 Model Comparison

For the *n*-processor VAXcluster system, the CTMC model faces largeness problem in respect to both generating state space and solving the model when *n* becomes large. The SRN model avoids the difficulty in generating state spaces and maintains solution accuracy. Using the MFTs model in the top level and using the CTMC sub-model for each processor in the lower level provides an efficient approximation method to analyze the system availability. In this section, the SRN model and the MFTs model are compared with respect to both solution accuracy and efficiency. Figure 12 shows the downtime per year computed by these two models. The results for the CTMC model are not included due to the complexity to construct the model when *n* is larger than 2. We can see that nearly the same downtime is obtained from these two models. How-

ever, the SRN model consumes more time to solve than the MFTs model. Figure 13 presents the log value of the execution time T (second) for these models.

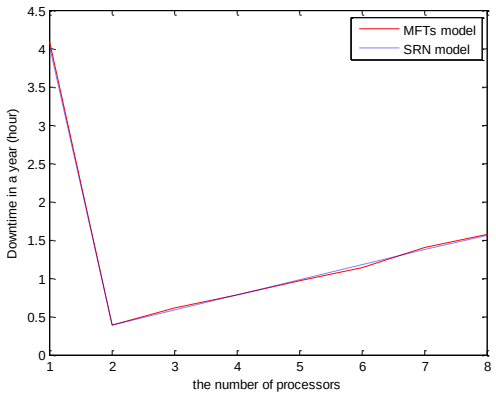


Fig. 12. Downtime Vs. the number of processors for different models

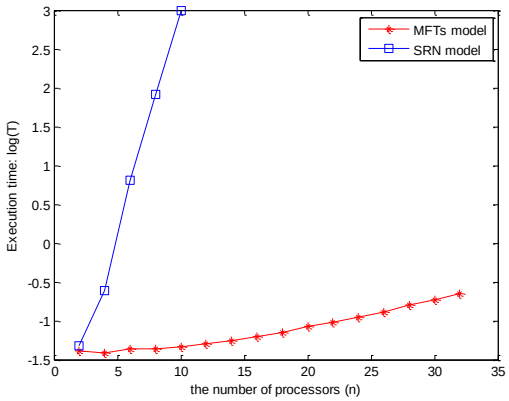


Fig. 13. Execution time Vs. the number of processors for different models

From the above results, we can see that the execution time for the MFTs model slightly increases as n increases. Regarding to the SRN model, the execution time exponentially increases Therefore, we can conclude that he MFTs model for VAX-cluster analysis is more efficient, regarding both constructing the model and solving the model, than the CTMC model and the SRN model while still maintaining the accuracy of the results.

4 Conclusions

There are many systems which have the multi-state. We have presented usages of three analytic modeling techniques (CTMC, SRN and MFTs) to evaluate the availability of multi-state systems. We have shown the two multi-state systems (the two boards system and the processors subsystem of the VAXcluster). We have compared three modeling techniques in terms of model accuracy and execution time. We have also shown the optimal number of processor for the processor subsystem of the VAXcluster.

References

1. Ibe, O. C., Howe, R. C., Trivedi, K. S.: Approximate Availability Analysis of VAXcluster Systems. IEEE Transactions on Reliability. 38(1) (1989) 146–152
2. Muppala, J., Sathaye, A., Howe, R., Trivedi, K. S.: Dependability Modeling of a Heterogenous VAXcluster System using Stochastic Reward Nets, In: Avresky, D.R: Hardware and Software Fault Tolerance in Parallel Computing Systems, Ellis Horwood Ltd., (1992) 33–59
3. Trivedi, K. S.: Probability and Statistics with Reliability, Queuing and Computer Science Applications. 2nd edn. John Wiley and Sons, New York (2001)
4. Trivedi, K. S., Sahner, R.: SHARPE at the age of twenty two. ACM SIGMETRICS Performance Evaluation Review. 36(4) (2009) 52–57
5. Zang, X., Wang, D., Sun, H., Trivedi, K. S.: A BDD-Based Algorithm for Analysis of Multistate Systems with Multistate Components. IEEE Transactions on Computers 52(12) (2003) 1608–1618