Largeness Avoidance in Availability Modeling using Hierarchical and Fixed-point Iterative Techniques

Harish Sukhwani[1], Andrea Bobbio[2], and Kishor S. Trivedi*[1]
[1]*Duke University, Durham, U.S.A.*
[2]*Università del Piemonte Orientale, Alessandria, ITALY*

(Received on December 25, 2014 , revised on March 05, 2015)

Abstract: Accurate modeling of availability is a practical problem in today's complex high-availability systems. But as the system gets more complex, the state-space required for accurate modeling tends to grow very fast. In order to mitigate the largeness in model generation / solution, the system model could be divided into subsystem models, and solution for sub-models can be combined to yield overall model solution. Such hierarchical composition techniques reduce the state-space tremendously. But simple hierarchical techniques provide exact results only when sub-model solutions are independent. In many scenarios, some components or procedures are shared across subsystems, which violate independence in sub-model solution. Hence approximation techniques like nearly independent systems are required to model systems where sub-model solutions are dependent. This paper demonstrates approximation techniques for availability modeling for a fluid pressure control system using the concepts of nearly independent subsystems and fixed-point iteration.

Keywords: *Availability, fixed-point iteration, hierarchical modeling, largeness avoidance, Markov models*

1. Introduction

As the human race is evolving, we are increasingly dependent on the technological infrastructure like the internet, transportation, energy supply, communications, health, financial services and commerce. Even short outages in such infrastructure can cause economic losses, environmental problems or, in the worst case, human life loss. Thus it is imperative for such critical systems and infrastructure to be designed and implemented with assurance of high reliability and availability.

Large fault tolerant systems consist of many subsystems, where each subsystem has many components. During the operation of complex systems, failure and repair of individual components or subsystems is inevitable. Also high availability requirements necessitate very accurate system models [1]. Thus availability modeling is a practical problem for evaluating dependability of such fault tolerant systems.

State-based methods are widely used for dependability and performance modeling of complex systems. These techniques are able to capture dynamic behavior and dependencies that non-state-space methods like reliability block diagrams (RBD) and fault trees cannot capture [2, 3, 4]. However, state-space models suffer from state-space explosion *i.e.*, extremely large state-space is required for the accurate modeling of real systems, referred to as *largeness* [3]. Two general approaches for dealing with largeness are *largeness avoidance* and *largeness tolerance*.

In *largeness tolerance* [5] techniques, system is modeled using higher level paradigms like stochastic Petri-nets which provide a compact representation of a model, which is automatically converted into an underlying Markov model. Special algorithms and data structures can be utilized to manipulate and store the underlying CTMCs thus reducing the space requirements of the state space, generator matrix, and iteration vectors [6].

*Corresponding author's email: ktrivedi@duke.edu

Petri nets are particularly suited to represent logical interactions among parts of the system in a natural way. It is a powerful paradigm to represent situations involving synchronization, sequencing, concurrency and conflict. It is also useful to automate the generation of large state spaces. A Petri net consists of *places*, *transitions*, *arcs* and *tokens*. The vector of the number of tokens in all the places represents the marking of the Petri net, which is equivalent to a state in the underlying CTMC. When a transition fires, token(s) is removed from the input place and token(s) are added to the output places. If the transition firing times are exponentially distributed, such nets are called stochastic Petri nets (SPN), where the underlying reachability graph representing transitions from one marking to another is a homogeneous CTMC. The flavor of SPN that we use in this paper is called stochastic reward nets (SRN) [7], using package stochastic Petri net package (SPNP) [8]. Other flavors of SPN include generalized stochastic Petri nets (GSPN) [9] and Stochastic activity networks [10].

Another way to tackle largeness is to avoid generating a large model, called *largeness avoidance* [2, 3, 5]. Some of the techniques include state truncation methods, hierarchy, and fixed-point iteration. In our work, we focus on hierarchical modeling techniques [2, 3, 11], where a large system can be viewed as composed of subsystems, each of which in turn can be further sub-divided into smaller subsystems or basic components. Thus, the whole system forms a *hierarchy* of subsystems, which at the lowest level is composed of components. Instead of generating and solving the full model, if smaller models are generated whose solution is combined (or rolled up) to yield the overall model solution, it could alleviate the largeness problem significantly. These techniques are referred to as *hierarchical composition*, where the overall system model consists of one or more sub-models of possibly different types that interact with each other [12, 11]. The model solution is obtained by solving the sub-models and combining sub-model solutions.

If there are no shared components or procedures across subsystems, the availability of each subsystem is independent of each other, and the overall system availability can be computed from the availabilities of the individual subsystems. However, in real systems, some components or procedures are shared across subsystems and state-space models are needed to capture such interactions or dependencies. For example, if there are shared repair facilities across subsystems, the system availability cannot be computed by invoking the independent assumption. Although full system model can capture all such dependencies, generating and solving large monolithic models can be error prone or even impossible. Instead we can use approximation techniques to capture dependencies across sub-models, yet retain the hierarchical nature of the model.

The objective of this paper is to demonstrate the approximation techniques for analyzing large systems using the concepts of *nearly independent subsystems* and *fixed-point iteration* from [13]. The system under consideration is a repairable fluid pressure control system with four subsystems. The paper analyzes three possible system configurations combined with two different repair policies: independent repair (as many repair persons as failed subsystems) and shared preemptive repair policy with a single repair person and priority list. More complex non-preemptive repair policies are the object of ongoing research. In Section 2, subsystems have no redundancy, next each subsystem has redundant components (in Section 3), and finally the travel time of the repair-person is considered as well (in Section 4). For each scenario we compare the system steady-state availability with independence assumption, the monolithic solution over the whole state space (when feasible) and the approximate solution using a nearly independent assumption. Scalability of the nearly independent solution is analyzed vs system largeness in Section 5.

2. System Description: No Redundancy

Consider a system used to maintain the fluid pressure in a tank at a constant value. The system is represented in Figure 1 and is composed of four components (or blocks): a pressure sensor (**S**), a control logic (**L**), a group motor-pump (**P**) and a valve (**V**). The components are functionally connected in series since the correct operation of each component is needed to guarantee the correct functioning of the system.

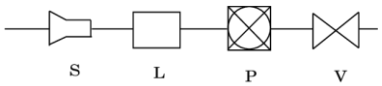


Figure 1: Series System for Guaranteeing a Constant Pressure in a Tank

Assume that each subsystem in Figure 1 consists of a single binary repairable component. Throughout the paper we assume that failure and repair times are exponentially distributed random variables with parameter values given in Table 1. If the components fail and get repaired independently, the resulting CTMC has $2^4 = 16$ states. However, we can exploit the total independence across components and use simple hierarchical solutions. Let $\lambda_S, \lambda_L, \lambda_P, \lambda_V$ be the failure rates and $\mu_S, \mu_L, \mu_P, \mu_V$ be the repair rates for the four components respectively. The steady-state availability for each component, whose value is reported in the last column of Table 1, can be obtained by the expression $A_i = \frac{\mu_i}{\lambda_i + \mu_i}$ (with $i = S,L,P,V$).

Thus we do have a hierarchical model with the top level being a series RBD and at the bottom level we have four 2-state CTMCs as shown in Figure 2. Under the independence assumption, the system steady-state availability A_{Sys} becomes

$$A_{Sys} = A_S A_L A_P A_V = 0.995754485$$

and can alternatively be computed from the monolithic CTMC (in Figure 3a). In this CTMC, the state name represents a vector of components (S, L, P, V) in UP (1) or DOWN (0) state. Also the states which have at most one change in the state name vector are connected to each other.

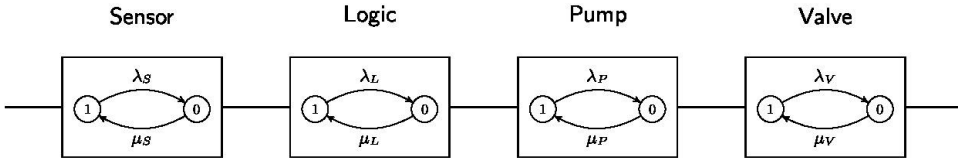
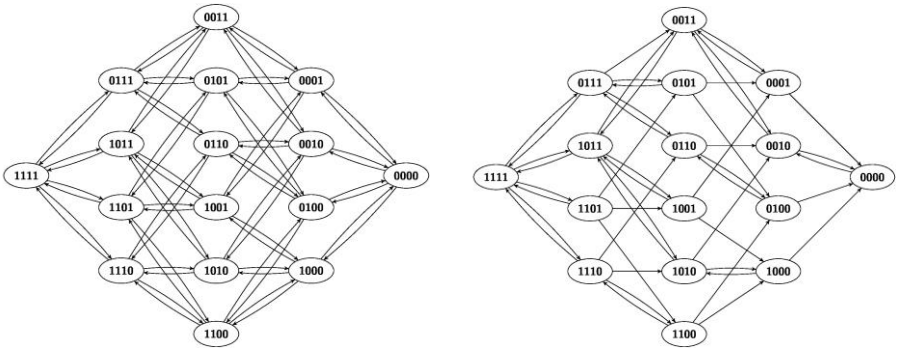


Figure 2: Reliability Block Diagram with 2-state CTMCs in each Block



(a) Independent Repair (b) Shared Repair: decreasing *MTTR* priority

Figure 3: CTMC for System with no Component-level Redundancy

2.1 Shared Repair-Person: Exact Solution

Now suppose we wish to consider a situation in which a single repair-person is shared among the components. If more than one component is failed, the repair-person must decide which component to repair first. Different preemptive and non-preemptive repair policies can be envisaged and they can be easily represented as a CTMC on the whole state space. By adopting a *preemptive repair priority policy*, a priority list is defined and the repair crew repairs the components according to the order defined in the priority list. Throughout the paper, the list is ordered according to the decreasing values of the *MTTR*, *i.e.*, the component with highest *MTTR* is repaired first (which is $P \succ V \succ S \succ L$). The CTMC with shared preemptive repair priority policy is shown in Figure 3b. With the parameters in Table 1, the steady-state availability is

$$A_{Sys} = 0.993414494 \tag{1}$$

Table 1: Failure, Repair Rates and Steady-State Availability for System with no Component-Level Redundancy

Component i	*MTTF* $\frac{1}{\lambda}$(hours)	$MTTF^{-1}$ λ	*MTTR* $\frac{1}{\mu}$ (hours)	$MTTR^{-1}$ μ	Availability $\frac{\mu}{\lambda+\mu}$
P	5000	2e-4	16	0.0625	0.99681021
V	10000	1e-4	8	0.125	0.99920061
S	2500	4e-4	4	0.25	0.99840260
L	2000	5e-4	2	0.5	0.99900100

2.2 Shared Repair-Person: Approximate Solution

The shared repair facility prevents the use of the independence assumption. An approximate hierarchical solution may be obtained by resorting to the concept of *nearly independent subsystems* developed in [14, 13]. A system is composed by nearly independent subsystems if the probability of a state of the overall system can be approximated by the product of probabilities of the states of individual subsystems

$$P(S = s_1, s_2, \dots, s_n) \approx P(S_1 = s_1)P(S_2 = s_2) \dots P(S_n = s_n)$$

where $P(S = s_1, s_2, \dots, s_n)$ is the probability of a state of the system where subsystem i is in state $S_i = s_i$. A system with dependent subsystems can be treated as *nearly independent*

if each individual subsystem can be modified to individually account for the effect of the dependence.

Consider the shared repair with priority $P \succ V \succ S \succ L$. If P is repaired first, the repair of the other subsystems with lower priority is delayed. We can account for this delay modifying the repair rates of the subsystems with lower priority. The CTMC M_P related to model P is solved first and the probability Q_P that the subsystem P is under repair is calculated. Thus the probability that the repair-person is idle is $q_P = (1 - Q_P)$. Subsystem V can be repaired only if the repair-person is not busy with subsystem P, which can be approximately accounted for by reducing the repair rate of subsystem V by the factor q_P:

$$\mu'_V = \mu_V q_P = \mu_V(1 - Q_p)$$

Table 2: Nearly Independent Approximation for Shared Repair for System with no Component-Level Redundancy

Sub-system i	Up states A_i	Repair states Q_i	Updated repair rates μ'_i	Indep. Repair Availability $A_{Indep.\ Repair}$	Shared Repair Availability $A_{Shared\ Repair}$
P	π_1	π_0	μ_P	0.996810207	0.996810207
V	π_1	π_0	$\mu_V(1 - Q_P)$	0.999200639	0.999198084
S	π_1	π_0	$\mu_S(1 - Q_P)(1 - Q_V)$	0.998402556	0.998396168
L	π_1	π_0	$\mu_L(1 - Q_P)(1 - Q_V)(1 - Q_S)$	0.999000999	0.998995392

Applying this idea iteratively, in order to start repair on subsystem M_i, the repair-person must be idle on all the subsystems M_j with $(j \leq (i - 1))$. This acyclic dependency between the subsystem models for repair priority $P \succ V \succ S \succ L$ is represented by the import graph shown in Figure 4. The updated repair rates for each subsystem are reported in the fourth column of Table 2.

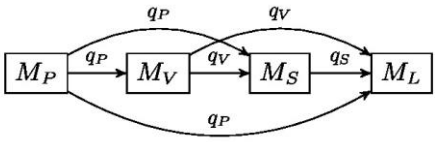


Figure 4: Import graph showing acyclic interaction between the sub-models

Comparing the last two columns of Table 2, we see that the difference between the exact and the approximate availability values increases as the priority decreases. The system steady-state availabilities for the three considered cases are reported in the first row of Table 5.

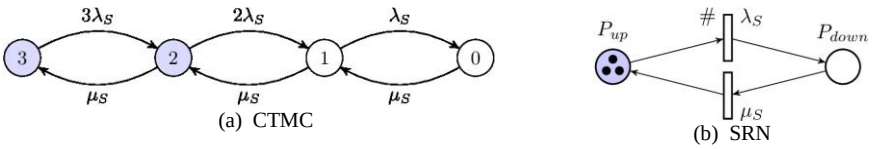
3. Redundant Components in Subsystems: Independent Repair

In order to increase the overall system availability, redundant repairable subsystems are introduced whose configurations are shown using both CTMC and equivalent stochastic reward net (SRN). For CTMCs, the UP states are denoted in dark shade. For SRNs, the rates and enabling functions are provided wherever applicable. The reward rates are skipped for brevity and can be understood from Eqn. (3).

Subsystem S - There are 3 sensors that work in a 2:3 logic with a single repair-person. The system is up when at least 2 out of 3 sensors are up. The CTMC and the SRN for the sensor block are shown in Figure 5. Note that although the system is considered down

in state 1 of the CTMC, one working sensor can fail. Thus the steady-state availability is given by:

$$A_S = \pi_3 + \pi_2$$

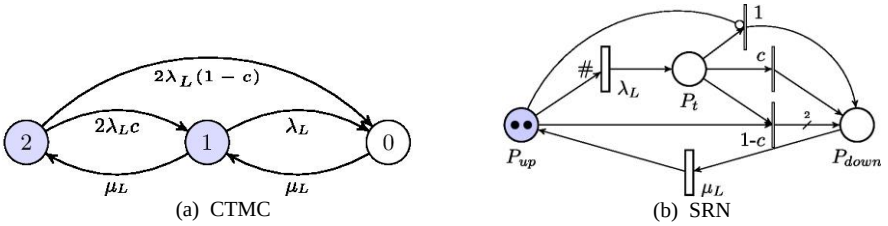


(a) CTMC (b) SRN

Figure 5: The 2:3 Sensor Block with Single Repair-Person

Subsystem L - The control logic is duplicated and the recovery mechanism has a coverage probability *c*. The CTMC and the SRN for the control logic block are shown in Figure 6, and its steady-state availability is given by:

$$A_L = \pi_2 + \pi_1$$

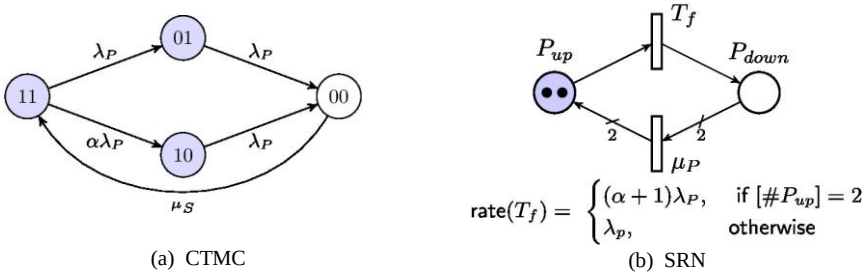


(a) CTMC (b) SRN

Figure 6: The Duplicated Control Logic with Coverage c

Subsystem P - The pump system is duplicated in warm stand-by configuration [15], and the block is repaired at once upon system failure. α is the dormancy factor for the standby unit. The CTMC and the SRN (where states (01) and (10) are merged) for the pump subsystem are shown in Figure 7, and its steady-state availability is given by:

$$A_P = \pi_{11} + \pi_{01} + \pi_{10}$$

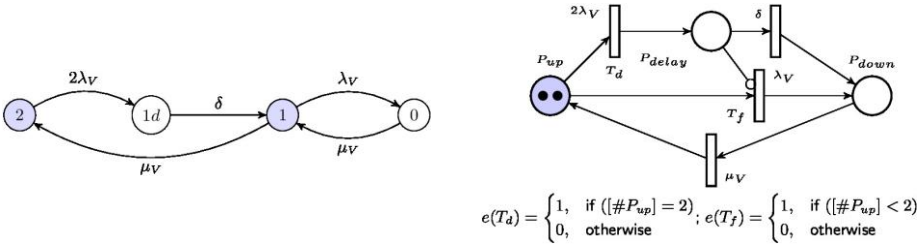


(a) CTMC (b) SRN

Figure 7: The Duplicated Pump Subsystem in Warm Stand-by Configuration

Subsystem V - The valve subsystem is duplicated with a single repair-person, but the reconfiguration upon failure of one component takes an exponentially distributed time with rate δ. During reconfiguration the valve subsystem is not available. The CTMC and the SRN for the valve subsystem are shown in Figure 8, and its steady-state availability is given by:

$$A_V = \pi_2 + \pi_1$$



(a) CTMC (b) SRN

Figure 8: The Redundant Valve Subsystem with Recovery Delay δ

If the subsystems are independently repairable, the steady-state availability of the whole system can be computed as the product of the availabilities of each subsystem. Thus we have a hierarchical model with the top level being a series RBD and at the bottom level we have four CTMCs as shown in Figure 9. In the computation, we assume the following additional numerical values: Dormancy factor α = 0.9 in Figure 7, coverage factor c = 0.9 in Figure 6 and delay δ = 1 in Figure 8. The overall system steady-state availability computed with the SHARPE [3] package becomes

$$A_{Sys} = A_S A_L A_P A_V = 0.997490970 \tag{2}$$

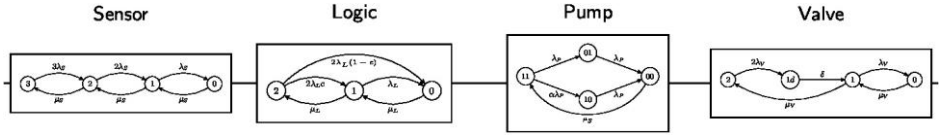


Figure 9: Fluid Pressure Control System as a RBD with Subsystems as Individual Blocks

3.1 Shared Repair-Person: Exact Solution

Suppose now that a single repair-person is shared among all the subsystems, in the same preemptive priority order $P \succ V \succ S \succ L$. Since the subsystems S, L, P, V have 4, 3, 4, 4 states respectively, the monolithic CTMC for the whole system has n = 4∗3∗4∗4 = 192 states. Generating such a CTMC can be cumbersome and error prone, hence the state space can be generated using SRN [7] as shown in Figure 10. The system is said to be available when there is at least two tokens in *UP* place for sensor block, and at least one token each in *UP* places for both logic and pump block, and at least one token in *UP* place & no token in *delay* place for valve block. Thus the reward rate can be defined as

$$r = \begin{cases} 1, & \left([\#P_{up}]_{\text{sensor}} \geq 2\right) \,\&\, \left([\#P_{up}]_{\text{logic}} \geq 1\right) \,\&\, \left([\#P_{up}]_{\text{pump}} \geq 1\right) \\ & \quad \&\, \left([\#P_{up}]_{\text{valve}} \geq 1 \,\&\, [\#P_{delay}]_{\text{valve}} = 0\right) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Using the same numerical values, the steady-state availability of this system obtained using SPNP is

$$A_{Sys} = 0.997485390685 \tag{4}$$

which is slightly less than the availability in the independent repair case (Eqn. (2)).

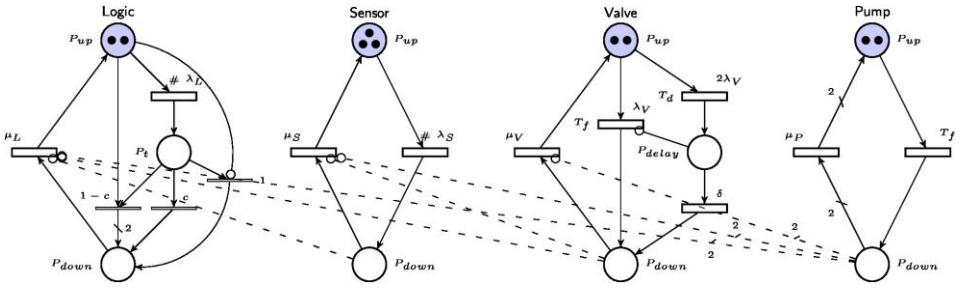


Figure 10: Modeling Repair Priority with Decreasing MTTR using SRN

3.2 Shared Repair-Person: Approximate Solution

A nearly independent approximation can be obtained similarly to Section 2.2. The explicit formulas for μ' for the ordered list $P \succ V \succ S \succ L$ and the availability results are given in Table 3. The whole system availability is $A_{Sys} = A_S A_L A_P A_V = 0.997489119$ which gives a slightly less optimistic value than the one from the monolithic exact model (in Eqn. (4)). Numerically, the availability is the same up to 5 digits of accuracy, but the modeling and computations are far less tedious compared to the full model constructed by SRN.

Table 3: Nearly Independent Approximation for Shared Repair for System with Redundancy

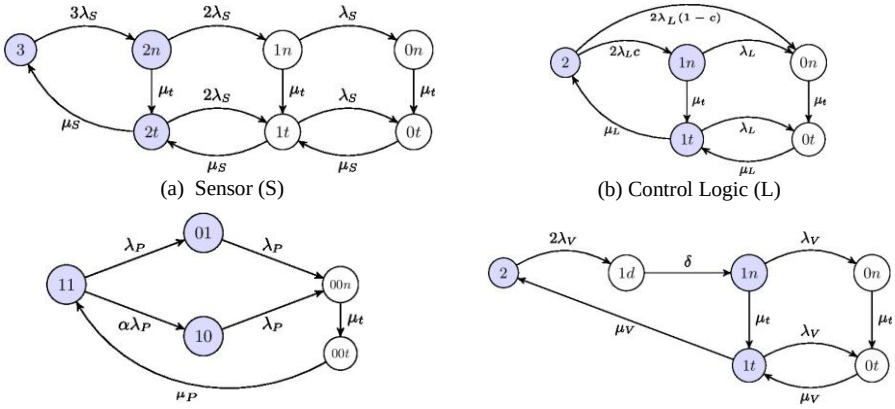
Sub-system i	Up states A_i	Repair states Q_i	Updated repair rates μ_i'	Indep. Repair Availability $A_{Indep.\ Repair}$	Shared Repair Availability $A_{Shared\ Repair}$
P	$\pi_{11} + \pi_{01} + \pi_{10}$	π_0	μ_P	0.997907835	0.997907835
V	$\pi_2 + \pi_1$	$\pi_1 + \pi_0$	$\mu_V(1 - Q_P)$	0.999799082	0.999799077
S	$\pi_3 + \pi_2$	$\pi_2 + \pi_1 + \pi_0$	$\mu_S(1 - Q_P)(1 - Q_V)$	0.999984689	0.999984576
L	$\pi_2 + \pi_1$	$\pi_1 + \pi_0$	$\mu_L(1 - Q_P)(1 - Q_V)(1 - Q_S)$	0.999798444	0.999796706

4. Travel Time for Repair-Person: Independent Repair

In a more realistic model the repair person must be notified as soon as his/her service is required, and if he/she is not on site, the *travel time* must be accounted for [13]. The travel time accounts for the delay occurring from the instant at which the request for service is notified to the instant at which the repair person is on site and ready to start the repair operation. Note that the travel time is often much larger than the proper repair time. We have further assumed that the travel time is exponentially distributed with rate μ_t.

To account for the travel time, we must duplicate all the states in which repair service is required, to distinguish whether the repair-person is already on site or not. The individual CTMCs for each subsystem are drawn in Figure 11, where we have denoted by a subscript *n* the states in which the repair-person is not on site and by a subscript *t* the states in which the repair-person is on site. From the states labeled *t*, when the repair person is on site, the repair is accomplished with the given repair rate, while from the states labeled *n* the travel transition with rate μ_t is introduced.

A hierarchical model with independent sub-models (similar to Figure 9) is used to compute the steady-state availability. Using SHARPE: $A_{Sys} = A_S A_L A_P A_V$ = 0.997255093



(a) Sensor (S) (b) Control Logic (L)

c) Motor pump (P) d) Valve (V)

Figure 11: CTMCs for All Subsystems including Travel Time with Shared Repair

4.1 Travel Time with Shared Repair-Person: Exact Solution

If a single repair-person is shared among all the subsystems with the same preemptive priority order, the monolithic CTMC model has 5∗6∗5∗7 = 1050 states, and it is cumbersome and error prone to generate.

Extending the model explained in Section 3.1, we use the SRN in Figure 12. Here an extra subnet is added to indicate the presence or absence of a repair-person.

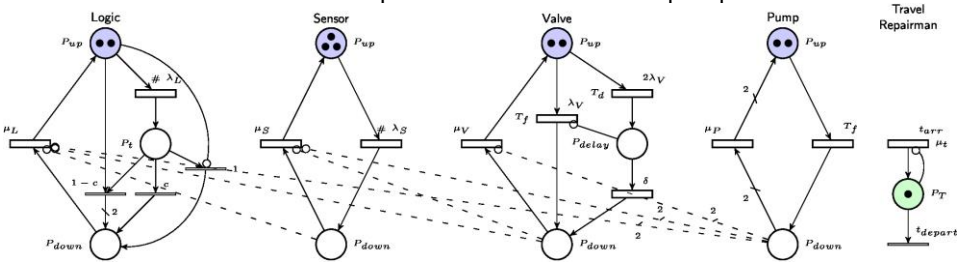


Figure 12: Modeling Repair-Person Arrival with Repair Priority using SRN

The transition t_{arr} is enabled when any of the subsystems is needing repair. Then the repair-person arrives in time exponentially distributed with rate μ_t. The immediate transition t_{depart} is enabled when the repairs for all the subsystems are completed. Thus the enabling condition for transition t_{arr} and t_{depart} can be defined as

$$e(t_{\text{arr}}) = \begin{cases} 1, & \left([\#P_{down}]_{\text{sensor}} \geq 1\right) \parallel \left([\#P_{down}]_{\text{logic}} \geq 1\right) \parallel \left([\#P_{down}]_{\text{valve}} \geq 1\right) \\ & \parallel \left([\#P_{down}]_{\text{pump}} \geq 2\right) \\ 0, & \text{otherwise} \end{cases}$$

$$e(t_{\text{depart}}) = \begin{cases} 1, & \left([\#P_{down}]_{\text{sensor}} = 0\right) \,\&\, \left([\#P_{down}]_{\text{logic}} = 0\right) \,\&\, \left([\#P_{down}]_{\text{valve}} = 0\right) \\ & \,\&\, \left([\#P_{down}]_{\text{pump}} = 0\right) \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

The monolithic CTMC can be automatically generated from the SRN to provide a value of the steady-state availability equal to A = 0.997431402583

4.2 Travel Time with Shared Repair-Person: Approximate Solution

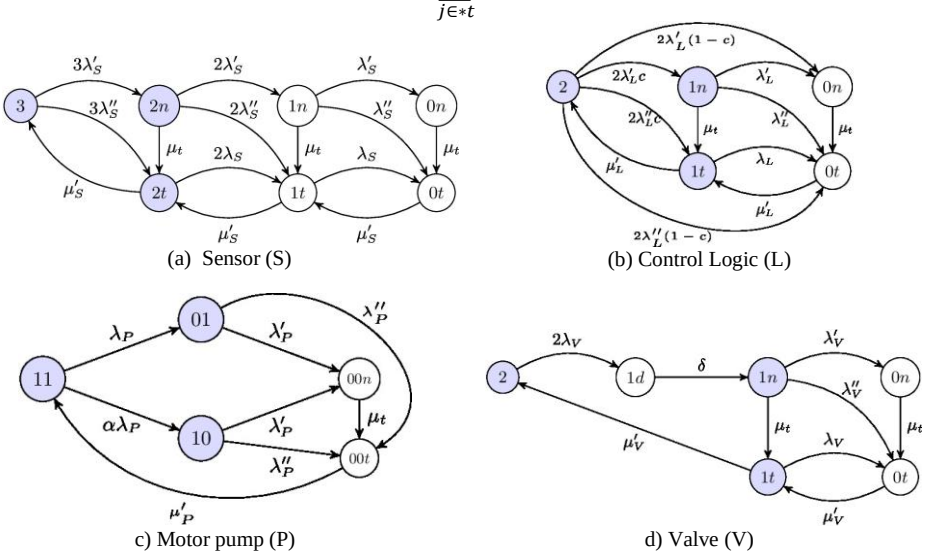
Let us attempt to find an approximate hierarchical solution by considering the subsystems as *nearly independent*. Compared to the previous example, the effect of sharing the travel time is more complex, as represented in the individual CTMC of Figure 13.

When a failure occurs in subsystem M_i, the following situations may occur:

1. In the states labeled *t*, the repair-person is on site and actually repairing a component in subsystem M_i; hence, the system may undergo a failure at rate λ_i or a repair with a repair rate μ_i' reduced by the priority order.
2. The repair-person is not on site M_i, hence the CTMC is in a state labeled *n* and the repair cannot start. Two situations may occur:
   (a) The repair-person is not on site and must be notified. The failure rates in this case are denoted by λ' and the failure transition is directed to another state *n*;
   (b) The repair-person is already on site in some subsystem $M_j \neq M_i$ and does not need to be notified. The failure rates in this case are denoted by λ'' and the failure transition is directed to a state labeled *t*.

For each of the subsystems *i*, the probability that the repair-person is on-site repairing the subsystem *i* is given by

$$r_i = \sum_{j \in *t} \pi_j$$



(a) Sensor (S) (b) Control Logic (L)

c) Motor pump (P) d) Valve (V)

Figure 13: CTMCs for all Subsystems including Travel Time with Shared Repair

The states $*n$ of the subsystem M_i indicate that the repair-person is not working on subsystem *i*. The outgoing failure rates from states $*n$ are split into two summands:

$$\lambda = \lambda' + \lambda'' \text{ where } \lambda' = \lambda \prod_{j \neq i} (1 - r_j) \text{ and } \lambda'' = \lambda \left[1 - \prod_{j \neq i} (1 - r_j) \right] \qquad (6)$$

The term $\prod_{j \neq i} (1 - r_j)$ in Eqn. (6) is the probability that the repair-person is not on site repairing any subsystem, and hence the subsystem *i* must notify the repair-person and

account for the travel time. Thus for evaluating the solution for each sub-model, there is a cyclic dependency with the solution from the other sub-models, as shown in the import graph in Figure 14. Here, the bold arrows indicate the transfer of parameter values in the current iteration, and the dotted arrows indicate the parameter values from the previous iteration. Using Brouwer's fixed-point theorem, the existence of a solution to fixed-point problem is proven in [13].

The CTMCs for the four subsystems are shown in Figure 13. As described in Section 4, model evaluation has dependencies for both failure rates λ's (as provided in Eqn. (6)), and repair rates μ's (as given in Table 4). The analysis (including the fixed-point iteration) is performed using SHARPE, and the results are reported in Table 4.

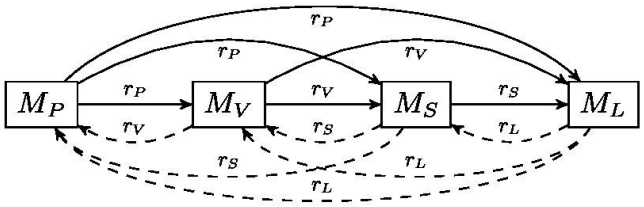


Figure 14: Import Graph showing Cyclic Interactions between the Sub-models

Table 4: Nearly Independent Approximation for Shared Repair with Travel Time for Repair-Person

Sub-system	Up states	Repair states	Updated repair rates	Indep. Repair Availability	Shared Repair Availability
i	A_i	Q_i	μ_i'	$A_{Indep.\ Repair}$	$A_{Shared\ Repair}$
P	$\pi_{11}+\pi_{01}+\pi_{10}$	π_{00t}	μ_P	0.997777365	0.997778487
V	$\pi_2+\pi_{1n}+\pi_{1t}$	$\pi_{1t}+\pi_{0t}$	$\mu_V(1-Q_P)$	0.999798942	0.999798938
S	$\pi_3+\pi_{2n}+\pi_{2t}$	$\pi_{2t}+\pi_{1t}+\pi_{0t}$	$\mu_S(1-Q_P)(1-Q_V)$	0.999979932	0.999979839
L	$\pi_2+\pi_{1n}+\pi_{1t}$	$\pi_{1t}+\pi_{0t}$	$\mu_L(1-Q_P)(1-Q_V)(1-Q_S)$	0.999697619	0.999696736

The steady-state availability is equal to A_{Sys} = 0.997255235, which is noticeably lower than case where repair-person travel time wasn't considered (in Section 3, Eqn. (4)). From the two rightmost columns in Table 4, it is interesting to note that the availability for each subsystem improves in presence of shared repair. This was unlike the case in Section 3.2, where shared repair slightly reduced the availability especially for lower repair-priority subsystems. Thus the overall availability in this scenario is higher compared to the independent repair-person case in Section 4.

Note that from Table 4, the subsystem *P* was clearly the bottleneck for the system availability. Since it has been given the highest preemptive repair priority, in cases where the repair-person is present for repairing other subsystems and subsystem *P* needs repair as well, a repair-person fixes both the subsystems in a single trip. Thus overall system availability went up even with a single repair-person, thus saving cost and efforts.

5. Numerical Results

Let us discuss the benefits of the approximation techniques based on two criteria: the accuracy of the approximation, and the reduction in computation costs due to the approximation. Table 5 summarizes the availability computation for the various system configurations and repair scenarios considered in the previous sections.

Table 5: Compare Availabilities of Different System Configurations and Repair Scenarios

Configuration	Indep. Repair	Shared Repair (exact)	Shared Repair (approx.)	# digits match
No Redundancy	0.995754485	0.993414494000	0.993414420	7
Redundancy (components)	0.997490970	0.997485390685	0.997489119	5
Redundancy + travel time (μ_t=0.1)	0.997255093	0.997431402583	0.997255235	3

Comparing the exact and approximate availability estimates in shared repair scenario, across all the configurations, the approximation techniques provide fairly good results, with match up to 5 digits of accuracy when redundancy of components is considered (2nd row of Table 5).

Let us compare the availability results between independent repair scenario and shared repair scenario to highlight some interesting results. In the configuration where travel time is considered (Section 4), notice that the system availability with shared repair-person is higher than that with individual repair-person per subsystem. This is in contrast with the scenario where repair-person was available on-site when subsystem component failure occurred (Section 3), where shared preemptive repair results in a decrease in availability. A possible explanation is that when the repair-person arrives to repair a subsystem, he/she would take care of the repairs for failures in other subsystems that are already down or whose failure occurs while the first repair is in process.

The above results could be sensitive to the mean time to travel for the repair-person. In Figure 15 we plot the difference between shared repair and independent repair availability (referred to as *Availability gap*) as a function of the mean time to travel ($1/\mu_t$). As the mean time to travel increases, the gap increases.

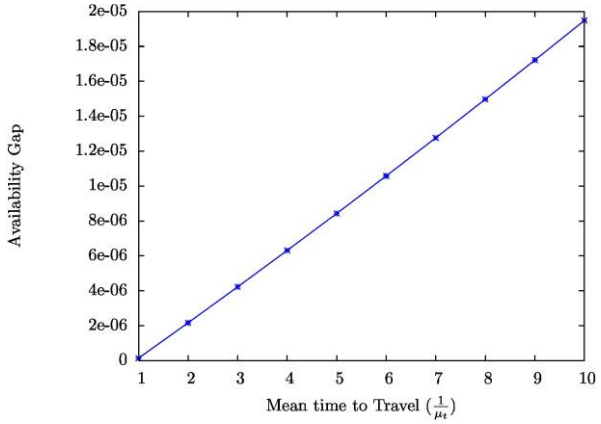


Figure 15: Availability Gap between Shared and Independent Repair for Different μ_t

Next, we compare the computation cost for the exact model with the approximate model. In order to deal with larger problem size, we consider system-level redundancy of say l, which means the entire system model is replicated l times in parallel, where all the systems share the same repair-person. The overall system is said to be available when any one of the l systems in parallel is available. Here, we consider the repair priority across systems first, and then within the same system. However, we have taken the configuration with no redundancy (as in Section 2), which means that the monolithic model for each system consists of 16 states.

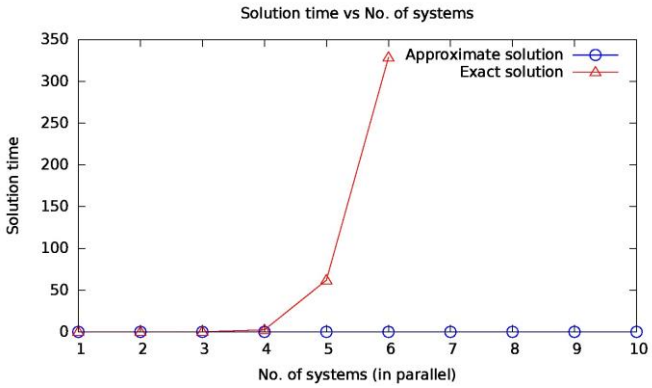


Figure 16: Solution Time as a Function of System Complexity for 'No Redundancy' Configuration

The plot in Figure 16 compares the time to generate and solve the exact SRN model with the approximation model using SHARPE. As the system size increases (shown in Table 6), the time required to solve the exact model seems to grow exponentially, but the time to solve the approximate model grows linearly. Also with exact SRN model it is possible to solve the model with 6 parallel lines only, before memory demands could not be met. Thus models for large fault-tolerant systems should be analyzed using approximation techniques.

Table 6: No. of States in underlying Markov Chain for Exact SRN Model and Approx. Model

No. of Systems	1	2	3	4	5
No. of States in Exact Model	16	256	4096	65536	1048576
No. of States in Approx. Model	16	32	48	64	80

6. Conclusion

This paper discusses nearly independent approximation techniques for availability modeling for a fluid pressure control system with two different repair policies: independent repair and preemptive repair priority policy. If a single repair-person present on-site repairs all subsystems with preemptive repair policy, it results in acyclic dependency among sub-model solutions. In this scenario, assuming reasonable parameters, with a shared preemptive repair, the steady-state availability is lower than the case where each subsystem had its own dedicated repair-person. If the repair-person's travel time is included in the model as well, it results in a cyclic dependency among sub-model solutions, which can be solved using fixed-point iteration. In this scenario, assuming reasonable parameters, with a shared preemptive repair, the steady-state availability is higher than the case where each subsystem had its own dedicated repair-person. This also saves cost and resources. Such intuitive (sometimes counter-intuitive) insights can be obtained from modeling large systems using hierarchical composition techniques with approximations. Future research on largeness avoidance in availability models will explore more complex repair policies with shared non-preemptive repair and non-priority repair order, like shared FCFS repair scheduling.

References

[1] Trivedi, K., R. Vasireddy, D. Trindade, S. Nathan, and R. Castro. *Modeling High Availability*, in Pacific Rim International Symposium on Dependable Computing (PRDC) 2006, Dec 2006: 154– 164.

[2] Trivedi, K. *Probability & Statistics with Reliability, Queueing & Computer Science Applications*, 2nd ed. Wiley, 2001.

[3] Sahner, R., K. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-based Approach Using the SHARPE Software Package*. Kluwer Academic Publisher, 1996.

[4] Evangelos, M., and P. Agapios. *Availability Assessment of Diesel Generator System of a Ship: A Case Study*. International Journal of Performability Engineering, 2013; 9(5): 561–567.

[5] Kaaniche, M., P. Lollini, A. Bondavalli, and K. Kanoun. *Modeling the Resilience of Large and Evolving Systems*. International Journal of Performability Engineering, 2008; 4(2):153–168.

[6] D. Nicol, W. Sanders, and K. Trivedi, "Model-based Evaluation: from Dependability to Security," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 48–65, Jan 2004.

[7] Muppala, J. K., G. Ciardo, and K. S. Trivedi. *Stochastic Reward Nets for Reliability Prediction*. In Communications in Reliability, Maintainability and Serviceability, 1994: 9–20.

[8] Ciardo G., J. Muppala, and K. Trivedi. *SPNP: Stochastic Petri Net Package*. In Proceedings International Workshop on Petri Nets and Performance Models - PNPM89. IEEE Computer Society, 1989:142–151.

[9] Marsan, M. A., G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*, 1st ed. Wiley, 1995.

[10] Meyer, J. F., A. Movaghar, and W. H. Sanders. *Stochastic Activity Networks: Structure, Behavior, and Application*. In International Workshop on Timed Petri Nets. IEEE Computer Society, 1985: 106–115.

[11] Lu, J., and S. S. Gokhale. *A Hierarchical Availability Analysis of Multi-tiered Web Applications*. International Journal of Performability Engineering, 2007; 3(3): 385 –387.

[12] Trivedi, K. S., and R. Sahner. *SHARPE at the Age of Twenty Two*. SIGMETRICS Perform. Eval. Rev., Mar. 2009; 36(4):52–57.

[13] Tomek, L., and K. Trivedi. *Fixed Point Iteration in Availability Modeling*. In Proceedings of the 5th International GI/ITG/GMA Conference on Fault-Tolerant Computing Systems, Tests, Diagnosis, Fault Treatment. London, UK. Springer-Verlag, 1991:229–240.

[14] Ciardo, G. and K. Trivedi. *A Decomposition Approach for Stochastic Reward Net Models*. Performance Evaluation, 1993; 18:37–59.

[15] Peng, R., Q. Zhai, L. Xing, and J. Yang. *Reliability of 1-out-of-(n+1) Warm Standby Systems subject to Fault Level Coverage*. International Journal of Performability Engineering, 2013; 9(1):117–120.

Harish Sukhwani is currently a Ph.D. student in Electrical & Computer Engineering at Duke University. He received his M.S. degree in Electrical Engineering from University of Southern California in 2010 and B.E. degree in Electronics & Telecommunication Engineering from University of Mumbai in 2007. Prior to joining the Ph.D. program, he worked in the industry for 2 ½ years as a Software Engineer for Cisco Systems in RTP, NC. His research interests are performance and dependability modeling, software reliability, software aging and rejuvenation.

Andrea Bobbio graduated from Politecnico di Torino in 1969. In 1971 he joined the Istituto Elettrotecnico Nazionale Galileo Ferraris di Torino, where he was involved in the organization of a ‘‘Reliability Club’’ (Circolo dell’Affidabilità). His activity then focused

on the modeling and analysis of the performance and reliability of stochastic systems. In 1992, he became an Associate Professor of Computer Engineering at the University of Brescia, and in 1995 he moved to the University of Torino. In 2000, he became a Full Professor at the Università del Piemonte Orientale of Alessandria, Italy. Bobbio has spent various research periods in well-recognized foreign universities. He has been principal investigator and leader of research groups in various research projects with public and private institutions, and he is author of several papers in international journals, conferences and workshops.

Kishor S. Trivedi holds the Hudson Chair in the Department of Electrical and Computer Engineering at Duke University, Durham, NC. He has a B.Tech. (EE) from IIT Mumbai and MS/PhD (CS) from the University of Illinois at Urbana-Champaign. He has been on the Duke faculty since 1975. He is the author of a well-known text entitled, Probability and Statistics with Reliability, Queuing and Computer Science Applications, originally published by Prentice-Hall; a thoroughly revised second edition (including its Indian edition) of this book has been published by John Wiley. He has also published two other books entitled, Performance and Reliability Analysis of Computer Systems, published by Kluwer Academic Publishers and Queueing Networks and Markov Chains, John Wiley. He is a Fellow of the Institute of Electrical and Electronics Engineers. He is a Golden Core Member of IEEE Computer Society. He has published over 500 articles and has supervised 45 Ph.D. dissertations. He is the recipient of IEEE Computer Society's Technical Achievement Award for his research on Software Aging and Rejuvenation. His research interests are in reliability, availability, performance and survivability of computer and communication systems and in software dependability. He works closely with industry in carrying our reliability/availability analysis, providing short courses on reliability, availability, and in the development and dissemination of software packages such as HARP, SHARPE, SREPT and SPNP.