PAPER Special Issue on Fault-Tolerant Computing

Three-Mode Failure Model for Reliability Analysis of Distributed Programs

Tatsuhiro TSUCHIYA†, Yoshiaki KAKUDA†, *and* Tohru KIKUNO†, *Members*

SUMMARY The distributed program reliability (DPR) is a useful measure for reliability evaluation of distributed systems. In previous methods, a two-mode failure model (working or failed) is assumed for each computing node. However, this assumption is not realistic because data transfer may be possible by way of a computing node even when this node can neither execute programs nor handle its data files. In this paper, we define a new three-mode failure model for representing such a degraded operational state of computing nodes, and present a simple and efficient analysis method based on graph theory. In order to represent the degraded operational state, a given graph expressing a distributed system is augmented by adding new edges and vertices. By traversing this augmented graph, the reliability measure can be computed. Examples show the clear difference between the results of our proposed method and those of the previous ones.

key words: *distributed system, distributed programs, reliability, 3-mode failure, file spanning tree*

1. Introduction

In distributed systems, the increase in reliability and fault-tolerance is achieved not only through the redundancy in computing nodes and communication links but also redundant distribution of programs and data files. In order to capture its effects, Kumar et al. proposed a reliability measure called *Distributed Program Reliability* (DPR) [8]. The DPR is defined as the probability that a given program can run while accessing all the required files in spite of failures among the nodes and the links.

Several methods of computing the DPR have been proposed [2], [3], [6]–[10]. These methods assume that each component is either fully operational or completely failed. However, the assumption is not realistic and results in an underestimate of reliability because data transfer may be possible by way of a computing node even when this node can neither execute programs nor handle its data files. The reason is that communication protocols usually have a layered structure. Among these layers, application programs and protocols for accessing remote files are working at the highest layer, while relay of data is done at lower layers.

For example, OSI reference model is composed of seven layers [5]. In this model, the function of each layer depends on its lower layers but not on its higher layers, and data transfer on intermediate nodes is performed by the third layer and its lower layers. Hence, even when intermediate nodes are not fully operational, two distinct operational nodes can communicate mutually via intermediate nodes if they can still correctly transfer data.

For exact evaluation, we introduce a new three-mode failure model for computing nodes in this paper. That is, we consider a degraded operational mode in addition to an operational mode and a failed mode. We assume that when a node is in this degraded operational mode, programs and files that reside at the node are not available, but data transfer via the node is possible.

To evaluate the DPR under the assumption of three-mode failure, we augment a given graph expressing a distributed system by new vertices and edges. Additionally, we introduce the notion of state for each vertex or edge, and assume that the state is either *up* or *down*. By translating the failure modes of the components into the states of vertices and edges in the augmented graph, we can compute the DPR based on graph theory and probability theory.

2. System Model

The topology of the distributed system is specified by an undirected graph $G = (V, E)$, where each vertex $x_i \in V$ represents a computing node and each edge $x_{i,j} \in E$ represents a bidirectional communication link between node x_i and node x_j. We assume that there exists no self loop in graph G. In the following, we denote $V = \{x_1, x_2, \cdots, x_{|V|}\}$ and $E = \{x_{a_1,b_1}, x_{a_2,b_2}, \cdots, x_{a_{|E|},b_{|E|}}\}$. The set of programs that can run on node x_i is denoted by PRG_i, while the set of files that are available at x_i is denoted by FA_i. The set of files required for successful execution of program P_i is given by FN_i. Figure 1 shows an example of distributed systems, where $FN_1 = \{F_1, F_2, F_3\}$.

As mentioned earlier, we assume a three-mode failure model for computing nodes in this paper. These three modes are defined as follows:

- Fully operational mode.
- Degraded operational mode: If a node is in this mode, programs cannot run on this node and files residing at the node cannot be accessed, but data transfer via this node is possible.

Manuscript received May 1, 1996.

†The authors are with the Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University, Toyonaka-shi, 560 Japan.

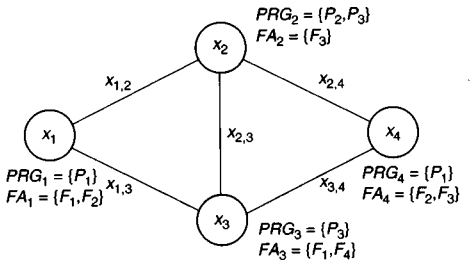


Fig. 1 Example of distributed systems.

- Completely failed mode.

On the other hand, for communication links, we assume a conventional two-mode failure model, i.e., each link is either operational or failed. Failures of nodes and links are assumed to be independent of others.

For each node x_i, the probability of being in the fully operational mode is given by p_i. Similarly, that of being in the degraded operational mode is given by q_i. For each link $x_{i,j}$, the probability of being operational is given by $r_{i,j}$. The previous model based on the assumption of two-mode failure is a special case of our model in which $q_i = 0$ for every node x_i.

Let $m_i \in \{f\_opr, d\_opr, fail\}$ denote the mode of node x_i, where $f\_opr, d\_opr, fail$ denote the fully operational mode, the degraded operational mode and the completely failed mode, respectively. Similarly, let $m_{i,j} \in \{opr, fail\}$ denote the mode of link $x_{i,j}$. Then, the status of the system S is represented by

$$S = (m_1, m_2, \cdots, m_{|V|}, m_{a_1,b_1}, m_{a_2,b_2}, \cdots, m_{a_{|E|},b_{|E|}}).$$

According to this failure model, we assume that program P_j can run successfully if and only if there is at least one tree $t = (V_t, E_t)$ of G such that all the following conditions are satisfied.

C1 For any node $x_i \in V_t$, its mode $m_i = f\_opr$ or d_opr.

C2 For any link $x_{i,j} \in E_t$, its mode $m_{i,j} = opr$.

C3 t includes a node in the fully operational mode that can execute P_j, i.e., for certain i, $P_j \in PRG_i$ and $m_i = f\_opr$.

C4 All required files can be provided by the fully operational nodes of t, i.e.,

$$FN_j \subset \bigcup_{\{x_i \in V_t | m_i = f\_opr\}} FA_i.$$

3. Augmenting the Original Graph

Under the assumption of three-mode failure, it is difficult to estimate the probability of a program being operational in a straight manner. To solve this problem, we augment the original graph $G = (V, E)$ and get a new graph $G' = (V', E')$ which can represent three-mode failure.

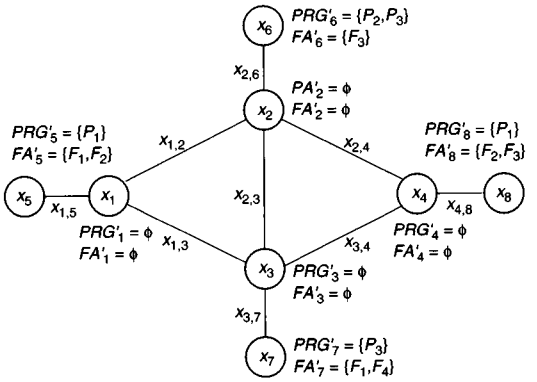


Fig. 2 Augmented graph G'.

The augmenting procedure is as follows: if node x_i neither holds any file nor is able to execute any program, the fully operational mode of x_i is not taken into account. On the other hand, if node x_i holds some files or can execute programs, three failure modes must be distinguished clearly. For each of such vertices, i.e., $x_i \in V$ such that $PRG_i \neq \phi$ or $FA_i \neq \phi$, we add a new vertex $x_{|V|+i}$ and a new edge $x_{i,|V|+i}$. Consequently we obtain a new undirected graph G'.

Intuitively, in the augmented graph G', vertex $x_i \in V'$ represents the data transfer function of node x_i, while vertex $x_{|V|+i} \in V'$ represents the function for handling data files and processing programs. As a result, node x_i in G is represented by tree $(\{x_i, x_{|V|+i}\}, \{x_{i,|V|+i}\})$ of G'. (Edge $x_{i,|V|+i}$ has no physical meaning.) Additionally, PRG'_i and FA'_i are defined for each vertex $x_i \in V'$ as follows.

$$PRG'_i = \begin{cases} \phi & i \leqq |V| \\ PRG_{i-|V|} & i > |V| \end{cases}$$

$$FA'_i = \begin{cases} \phi & i \leqq |V| \\ FA_{i-|V|} & i > |V| \end{cases}$$

For example, the graph illustrated in Fig. 1 is transformed as shown in Fig. 2.

By assigning either the up state or the down state to each vertex and edge of G', G' can represent the status of the system S. For this purpose, we associate functions $f_i(S)$ and $f_{i,j}(S)$ with each vertex $x_i \in V'$ and each edge $x_{i,j} \in E'$, respectively. These functions determine the states of all the components of G'.

$$\text{For } i \leqq |V|, \quad f_i = \begin{cases} up & m_i \in \{f\_opr, d\_opr\} \\ down & \text{otherwise} \end{cases}$$

$$\text{For } i > |V|, \quad f_i = \begin{cases} up & m_{i-|V|} = f\_opr \\ down & \text{otherwise} \end{cases}$$

$$\text{For } x_{i,j} \in E, \quad f_{i,j} = \begin{cases} up & m_{i,j} = opr \\ down & \text{otherwise} \end{cases}$$

$$\text{For } x_{i,j} \in E' - E, \quad f_{i,j} = up$$

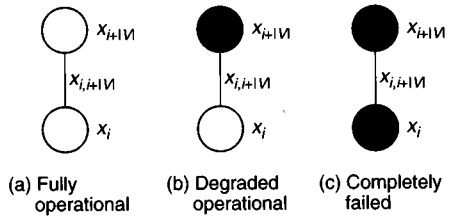


Fig. 3 Representation of three-mode failures.

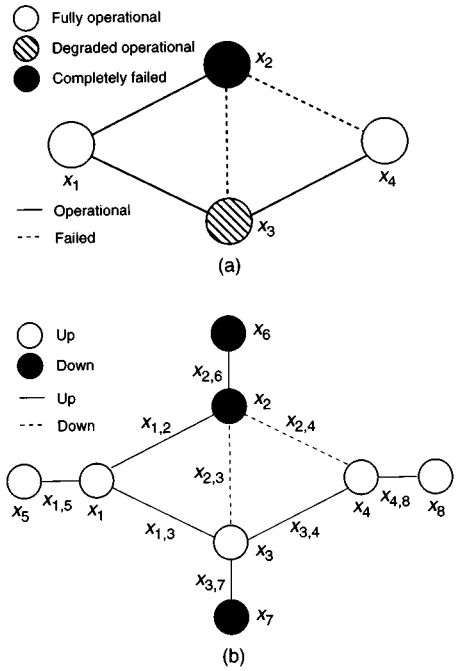


Fig. 4 (a) Status of the system, and (b) its representation.

According to these functions, the mode of node x_i is represented as shown in Fig. 3. In the figure, a white circle represents a vertex in the up state, while a black circle represents a vertex in the down state. A solid line connecting two circles represents an edge in the up state. Figure 4 (a) shows an example of the status of the distributed system in Fig. 1. Its corresponding representation in the augmented graph is shown in Fig. 4 (b), where an edge in the down state is denoted by a dotted line connecting two circles.

Lemma 1: There is a tree t of $G = (V, E)$ satisfying conditions C1, C2, C3 and C4 for program P_j if and only if there is a tree $t' = (V'_t, E'_t)$ of G' such that the following conditions hold.

C1' All vertices and edges of t' are in the up state.

C2' If t' includes vertex x_i with $i > |V|$, then t' also includes vertex $x_{i-|V|}$.

C3' t' includes a vertex where program P_j is available, i.e., for certain i, $P_j \in PRG'_i$.

C4' All the required files can be available at the vertices of t', i.e.,

$$FN_j \subset \bigcup_{x_i \in V'_t} FA'_i.$$

Proof: Assume that tree $t = (V_t, E_t)$ of G satisfies conditions C1, C2, C3 and C4. Then for any vertex $x_i \in V_t$ with $m_i = f\_opr$ such that $PRG_i \neq \phi$ or $FA_i \neq \phi$, both vertex $x_{i+|V|}$ in the up state and edge $x_{i,i+|V|}$ in the up state exist in G'. Let V_{add} and E_{add} be sets defined by

$$\begin{aligned} V_{add} &= \{x_{i+|V|} | x_i \in V_t \wedge m_i = f\_opr \\ &\quad \wedge (PRG_i \neq \phi \vee FA_i \neq \phi)\}, \quad \text{and} \\ E_{add} &= \{x_{i,i+|V|} | x_i \in V_t \wedge m_i = f\_opr \\ &\quad \wedge (PRG_i \neq \phi \vee FA_i \neq \phi)\}. \end{aligned}$$

By definition, for any $x_i \in V_{add}$, $PRG'_i = PRG_{i-|V|}$ and $FA'_i = FA_{i-|V|}$. Then, for certain $x_k \in V_{add}$, $PRG'_k = PRG_{k-|V|} \ni P_j$, and

$$\bigcup_{x_i \in V_{add}} FA'_i = \bigcup_{\{x_i \in V_t | m_i = f\_opr\}} FA_i \supset FN_j.$$

Hence, tree $(V_t \cup V_{add}, E_t \cup E_{add})$ is a tree of G' that satisfies conditions C1', C2', C3' and C4'.

Next, assume that tree $t' = (V'_t, E'_t)$ of G' satisfies conditions C1', C2', C3' and C4'. Let V'_{add} be the set of all vertices which are in V'_t but not in G, i.e.,

$$V'_{add} = \{x_i \in V'_t | i > |V|\}.$$

Then any vertex $x_i \in V'_{add}$ is adjacent only to $x_{i-|V|}$, and $x_{i-|V|}$ exists in t'. From this, a subgraph of t' which is obtained by removing all vertices in V'_{add} and their adjacent edges is a tree of G. Let T denote this tree, i.e., $T = (V'_t - V'_{add}, E'_t - E'_{add})$, where $E'_{add} = \{x_{i-|V|,i} | x_i \in V'_{add}\}$. Then, any components (nodes and links) of T is not in the completely failed mode. Moreover, since tree $(\{x_{i-|V|}, x_i\}, \{x_{i-|V|,i}\})$ with $x_i \in V'_{add}$ is up, computing node $x_{i-|V|}$ is fully operational. Hence, for certain $x_k \in V'_{add}$, $PRG_{k-|V|} = PRG'_k \ni P_j$, and

$$\bigcup_{\{x_i \in V'_t - V'_{add} | m_i = f\_opr\}} FA_i = \bigcup_{x_i \in V'_{add}} FA'_i \supset FN_j.$$

Hence, tree T is a tree of G that satisfies conditions C1, C2, C3 and C4. □

Consider tree $(\{x_1, x_3, x_4\}, \{x_{1,3}, x_{3,4}\})$ in Fig. 4 (a) as an example. Since this tree satisfies conditions from C1 through C4 for program P_1, it can enable execution of this program. This tree corresponds to tree $(\{x_1, x_3, x_4, x_5, x_8\}, \{x_{1,3}, x_{1,5}, x_{3,4}, x_{4,8}\})$ of G' shown in Fig. 4 (b), which satisfies conditions from C1' through C4'.

In this paper, we say that a subgraph of G' is *up* if and only if it satisfies condition C1', that is, all of its components are in the up state. From Lemma 1 and the assumption on program execution, when there is an up tree of G' satisfying conditions C2',C3' and C4' for a program, the successful execution of the program is

possible. We define a file spanning tree (FST) of the augmented graph G' as a tree of G' satisfying conditions C2', C3' and C'4. This definition is similar to the definition in the previous work, except that there is no condition corresponding to C2'. By Lemma 1 and the definition of FST, the following theorem is obtained.

Theorem 1: A program can run successfully if and only if there is at least one up FST for that program in the augmented graph G'.

As shown later, the DPR can be computed based on this Theorem 1 and probabilities of the components of G' being up. Let u_i be the event that vertex x_i is up and let $u_{i,j}$ be the event that edge $x_{i,j}$ is up, respectively. Then, these probabilities are obtained as follows.

$$\text{For} \quad i \leqq |V|, \quad \Pr(u_i) = p_i + q_i$$

$$\text{For} \quad i > |V|, \quad \begin{cases} \Pr(u_i | u_{i-|V|}) = \frac{p_i}{p_i+q_i} \\ \Pr(u_i | \overline{u_{i-|V|}}) = 0 \end{cases}$$

$$\text{For} \quad x_{i,j} \in E, \quad \Pr(u_{i,j}) = r_{i,j}$$

$$\text{For } x_{i,j} \in E' - E, \quad \Pr(u_{i,j}) = 1$$

4. Evaluation of Reliability

As mentioned earlier, the distributed program reliability (DPR) is defined as the probability that a given program can run successfully. According to Theorem 1, the DPR is written as follows.

$$\text{DPR} = \Pr(\text{For the program, at least one up FST exists in the augmented graph } G').$$

Like in [4], we introduce the notion of minimal file spanning tree (MFST) for efficient evaluation. An MFST is defined as an FST such that no FST which is a subgraph of that FST exists. Since we assume that each vertex or edge in G' is either up or down, an up MFST exists if an up FST exists. Its reverse also holds. Hence the DPR can be written as follows.

$$\text{DPR} = \Pr(\text{For the program, at least one up MFST exists in the augmented graph } G').$$

This probability can be computed by the following two steps:

Step 1. Find all MFST's for the program of interest in the augmented graph G'.

Step 2. Calculate the probability that at least one MFST is up by applying a terminal reliability algorithm. This probability is equal to the DPR.

4.1 Enumeration of MFST's (Step 1)

For MFST enumeration, we can apply an algorithm in [8], which is based on the two-mode failure model, with modification. The algorithm generates all the MFST's in nondecreasing order of their size, where the size is defined as the number of links in an MFST. In the two-mode failure model, possible sizes of MFST's range from 0 up to $|V| - 1$.

On the contrary, under our definition of MFST, any tree of size 0 never be an MFST because any tree consisting of only vertex x_i with $i > |V|$ does not satisfy C2', and any tree consisting of only vertex x_i with $i \leqq |V|$ satisfies neither C3' nor C4'. In addition, if a tree of G' does not satisfy C2', this tree is a vertex x_i with $i > |V|$ because x_i is adjacent only to $x_{i-|V|}$. In other words, any tree whose size is more than zero satisfies C2'.

Therefore if the previous algorithm is modified so as to begin with determining MFST's of size 1, it can be applied for enumeration of MFST's in G'. The algorithm is as follows.

```
/* Step 1.1: Initialization */
  TRY := ⋃_{x_i | P_j ∈ PRG'_i} tree ({x_i, x_{i+|V|}}, {x_{i,i+|V|}})
  FOUND := φ
/* Step 1.2: Generation of all MFST's */
  While (TRY ≠ φ) do
    /* 1.2.1 Checking step */
      For all t ∈ TRY do
        If (t has all needed files) Then
          If (t is not a superset of any tree in
              FOUND)
            Then
              add t to FOUND
              remove t from TRY
            Else
              remove t from TRY
          END_If
        END_If
      END_For
    /* 1.2.2 Expanding step */
      NEW := φ
      For all t = (V_t, E_t) ∈ TRY do
        AE := set of all adjacent edges to t
        For all (x_{i,j} ∈ AE, i ∈ V_t ∧ j ∉ V_t) do
          newt := (V_t ∪ {x_{i,j}}, E_t ∪ {x_j})
          add newt to NEW
        END_For
      END_For
      TRY := NEW
  END_While
```

4.2 Terminal Reliability Algorithms (Step 2)

Once all the MFST's have been found, the next step is to find the probability that at least one of them is up. In [8], [10], terminal reliability evaluation algorithms based on path enumeration, such as Abraham's one [1] and SYREL [4], are used for this purpose. From given subgraphs, these algorithms can derive the probability that at least one subgraph remains operational based on the two-mode failure model.

In the algorithms, it is also assumed that the event

of any component being operational is independent of those of other components. On the contrary, as shown in Sect. 3, the event that vertex x_i with $i > |V|$ is in the up state is dependent on the state of vertex $x_{i-|V|}$. In spite of this discrepancy, we can use the terminal reliability algorithms to compute the DPR without any modification. This reason is explained as follows.

Let $M = \{MFST_1, MFST_2, \cdots, MFST_{|M|}\}$ be the set of all MFST's, and let U_j be the event that $MFST_j$ is up. Clearly the following equation holds.

$$\mathrm{DPR} = \Pr(U_1 \vee U_2 \vee \cdots \vee U_{|M|})$$

Now, let x_i denote an arbitrary vertex which is in G' but not in G, and let M_{in} be the set of all MFST's including x_i. By definition, $i > |V|$, and any MFST in M_{in} also includes vertex $x_{i-|V|}$. For $MFST_j \in M_{in}$, let U'_j be the event that all components of $MFST_j$ except for x_i are in the up state. Note that u_i denotes the event that x_i is in the up state. Then, the DPR can be written as follow.

$$\begin{aligned}\mathrm{DPR} &= \Pr\left(\bigvee_{MFST_j \in M - M_{in}} U_j\right) \\ &\quad + \Pr\left(\bigvee_{MFST_j \in M_{in}} U_j \wedge \bigwedge_{MFST_j \in M - M_{in}} \overline{U}_j\right) \\ &= \Pr\left(\bigvee_{MFST_j \in M - M_{in}} U_j\right) \\ &\quad + \Pr\left(u_i \wedge \bigvee_{MFST_j \in M_{in}} U'_j \wedge \bigwedge_{MFST_j \in M - M_{in}} \overline{U}_j\right)\end{aligned}$$

Since any $MFST_j \in M_{in}$ includes $x_{i-|V|}$, whenever U'_j with $MFST_j \in M_{in}$ occurs, $x_{i-|V|}$ is in the up state. Hence, this equation can be transformed as follows.

$$\begin{aligned}\mathrm{DPR} = \Pr\left(\bigvee_{MFST_j \in M - M_{in}} U_j\right) + \Bigg\{ &\Pr(u_i | u_{i-|V|}) \\ &\cdot \Pr\left(\bigvee_{MFST_j \in M_{in}} U'_j \wedge \bigwedge_{MFST_j \in M - M_{in}} \overline{U}_j\right)\Bigg\}\end{aligned}$$

Clearly, both any $U_j(\overline{U_j})$ with $MFST_j \in M - M_{in}$ and any U'_j with $MFST_j \in M_{in}$ are independent of u_i because none of their corresponding subgraphs include x_i. The above equation implies that the DPR has no relation to the probability of x_i being up when $x_{i-|V|}$ is known to be down. Hence, even if $\Pr(u_i|\overline{u_{i-|V|}}) = \Pr(u_i|u_{i-|V|})(= \frac{p_i}{p_i+q_i})$, the DPR would not change. Notice that $\Pr(u_i|\overline{u_{i-|V|}}) = \Pr(u_i|u_{i-|V|})$ implies that u_i and $u_{i-|V|}$ are mutually independent. Thus, the terminal reliability algorithms can compute the DPR without any modification. (In the computation, we can choose $\Pr(u_i|u_{i-|V|})$ as the reliability of x_i with $i > |V|$.)

4.3 Time Complexity

Both in the conventional two-level failure model and in the proposed three-level failure model, the DPR can be computed by executing the corresponding MFST enumeration algorithm and a terminal reliability algorithm sequentially. In the two-level failure model, the first step is known to be dominant in terms of the time needed for DPR calculation [7], and this is true of the new failure model. This can be explained by the fact that in the first step $(|V|-1)^{(e-1)}$ intermediate trees are generated in the worst case, where e is the maximum degree of a vertex in G [7]. Hence, in the three-mode failure model, the number of generated intermediate trees is bounded by $(|V'|-1)^{(e'-1)}$, where e' is the maximum degree of a vertex in G'. When e' is fixed, since the number of intermediate trees is polynomial in terms of $|V'|$, the time complexity of the first step is also polynomial. Concerning terminal algorithms, their complexities are different with each other.

5. Numerical Examples

In this section, we present the results obtained by applying the proposed method to example systems. For simplicity, we assume here that for any $i, j, p_i = p_j, q_i = q_j$ and $r_{i,j} = 0.9$.

To evaluate the effects of the degraded operational mode, we define parameter α when $p_i < 1$ as follows.

$$\alpha = \frac{\Pr(\text{A node is degraded operational})}{\Pr(\text{A node is not fully operational})} = \frac{q_i}{1-p_i}$$

By definition, the value of α ranges from 0 to 1.0. Since q_i becomes zero if and only if $\alpha = 0$, the results obtained when $\alpha = 0$ are the same as those of the traditional two-level failure model. On the other hand, as the value of α becomes close to 1.0, the probability of being degraded operational, i.e., q_i becomes larger.

Example 1: Consider the simple system depicted in Fig. 1. We applied the proposed method to evaluate the DPR of program P_1 with $FN_1 = \{F_1, F_2, F_3\}$.

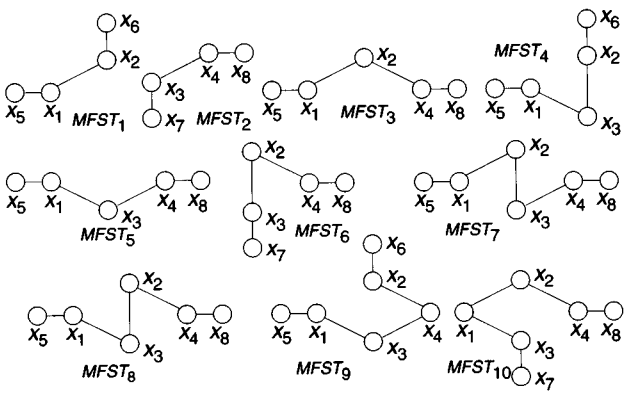


Fig. 5 All MFST's for program P_1 in Example 1.

Table 1 DPR of program P_1 in Example 1.

	$p_i = 0.80$	0.85	0.90	0.95	1.00
$\alpha = 0$	0.8405	0.8972	0.9442	0.9790	0.9989
0.25	0.8558	0.9083	0.9510	0.9819	
0.50	0.8691	0.9181	0.9572	0.9847	
0.75	0.8805	0.9266	0.9627	0.9874	

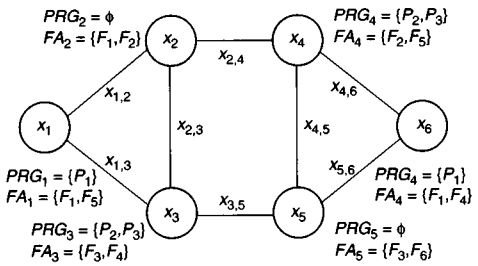


Fig. 6 Another example of distributed system.

Table 2 DPR of program P_1 in Example 2.

	$p_i = 0.80$	0.85	0.90	0.95	1.00
$\alpha = 0$	0.7993	0.8758	0.9378	0.9806	0.9995
0.25	0.8271	0.8943	0.9476	0.9838	
0.50	0.8497	0.9093	0.9557	0.9865	
0.75	0.8668	0.9209	0.9621	0.9887	

Step 1 enumerated 10 MFST's as shown Fig. 5. Using the terminal reliability algorithm in [1], Step 2 derived an expression corresponding to the DPR of program P_1 from the MFST's. The resultant expression is $\text{DPR} = 0.9p_i^2 + 0.9p_i^2(1-p_i-q_i) + 1.062p_i^2(p_i+q_i)(1-p_i-q_i) + 1.06929p_i^2q_i(p_i+q_i) + 0.98577p_i^2q_i^2 + 1.08549p_i^2q_i^2 + 0.09891p_i^4$. (The details of process for deriving the DPR is omitted here.) Table 1 shows the values of DPR with various p_i and α. This table implies that the previous methods result in an underestimate of DPR, and that considering the degraded operational mode of nodes is indispensable for exact evaluation. (Notice that the DPR with $\alpha = 0$ is equal to that in the conventional two-mode failure model.)

Example 2: Consider the distributed system shown in Fig. 6, where $FN_1 = \{F_1, F_2, F_3\}$. This example is also studied in [3] and [8]. Fifty four MFST's for program P_1 are found, and the values of DPR of P_1 shown in Table 2 are obtained. Compared with Example 1, the value of α affects the DPR more explicitly. The main reason is that the program is executed via a larger number of intermediate nodes than the previous example.

6. Conclusion

For exact reliability estimation, we have proposed a three-mode failure model for computing nodes in distributed systems. Based on this model, we present a method to compute the DPR. This method consists of three stages, i.e., augmenting the given graph that represents a system, enumerating MFST's, and computing the probability that one of them is up. Numerical examples show the clear difference between the estimation based on our model and that on the traditional two-mode failure model. As future work, we are planning to examine field data to obtain actual values of parameters, such as p_i and q_i.

References

[1] J.A. Abraham, "An improved algorithm for network reliability," IEEE Trans. Reliab., vol.R-28, no.1, pp.58–61, 1979.

[2] D.-J. Chen and M.-S. Lin, "On distributed computing systems reliability analysis under program execution constraints," IEEE Trans. Comput., vol.43, no.1, pp.87–97, 1994.

[3] D.-J. Chen and T.-H. Huang, "Reliability analysis of distributed systems based on a fast reliability algorithm," IEEE Trans. Parallel and Distributed Systems, vol.3, no.2, pp.139–154, 1992.

[4] S. Hariri and C.S. Raghavendra, "SYREL: A symbolic reliability algorithm based on path and cutset methods," IEEE Trans. Comput., vol.C-36, no.10, pp.1224–1232, 1987.

[5] ISO, "Open Systems Interconnection—Basic Reference Model," ISO 7498, 1984.

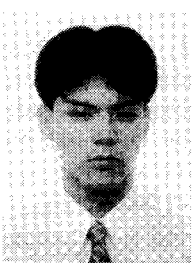
[6] A. Kumar and D.P. Agrawal, "A generalized algorithm for evaluating distributed-program reliability," IEEE Trans. Reliab., vol.42, no.3, pp.416–426, 1993.

[7] A. Kumar, S. Rai, and D.P. Agrawal, "On computer communication network reliability under program execution constraints," IEEE J. Select. Areas Commun., vol.6, no.8, pp.1393–1399, 1988.

[8] V.K.P. Kumar, S. Hariri, and C.S. Raghavendra, "Distributed program reliability analysis," IEEE Trans. Software Eng., vol.SE-12, no.1, pp.42–50, 1986.

[9] Noé Lopez-Bentiz, "Dependability modeling and analysis of distributed programs," IEEE Trans. Software Eng., vol.20, no.5, pp.345–352, 1994.

[10] C.S. Raghavendra, V.K.P. Kumar, and S. Hariri, "Reliability analysis in distributed systems," IEEE Trans. Comput., vol.37, no.3, pp.352–358, 1988.



Tatsuhiro Tsuchiya received the M.E. degree in computer science from Osaka University, in 1995. He is currently a research associate in the Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University. His research interests are in the areas of distributed and fault-tolerant systems and real-time systems.

Yoshiaki Kakuda received the B.S. degree in electronic engineering from Hiroshima University in 1978. He also received the M.S. and Ph.D. degrees in system engineering from the same university in 1980 and 1983, respectively. He is currently an associate professor in the Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University. His research interests include responsive systems and protocol engineering. He is a member of IEEE. He received the Telecom. System Technology Award from Telecommunications Advancement Foundation in 1992.

Tohru Kikuno received the B.E., M.S. and Ph.D. degrees in electrical engineering from Osaka University in 1970, 1972 and 1975, respectively. He joined Hiroshima University from 1975 to 1987. He is currently a professor in the Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University. His research interests include analysis and design of fault-tolerant systems, design of testing procedure of communication protocols and quantitative evaluation of software development processes. He is a member of IEEE and ACM. He received the Paper Award from Institute of Electronics, Information, and Communication Engineers of Japan in 1993.