PERGAMON

Computers & Operations Research 28 (2001) 625–635

computers & operations research

www.elsevier.com/locate/dsw

The distributed program reliability analysis on ring-type topologies

Min-Sheng Lin[a,*], Ming-Sang Chang[b], Deng-Jyi Chen[c], Kuo-Lung Ku[d]

[a]*Department of Information Management, Aletheia University, 32 Chen Li Road, Tamsui, Taipei, 25103 Taiwan, ROC*
[b]*Department of Information Technology, Chunghwa Telecommunication Training Institute, 168 Min Chu Road, Pan Chiao, Taipei, 22077 Taiwan, ROC*
[c]*Institute of Computer Science and Information Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsin Chu, Taiwan, ROC*
[d]*Chung-San Institute of Science and Technology, Tao-Yuan, Taiwan, ROC*

Abstract

Distributed computing system (DCS) has become very popular for its high fault-tolerance, potential for parallel processing, and better reliability performance. One of the important issues in the design of the DCS is the reliability performance. Distributed program reliability (DPR) is addressed to obtain this reliability measure. In this paper, we propose a polynomial-time algorithm for computing the DPR of ring topology and show that solving the DPR problem on a ring of trees topology is NP-hard. © 2001 Elsevier Science Ltd. All rights reserved.

Scope and purpose

The widespread use of distributed computing system is due to the price–performance revolution in microelectronics, the development of cost-effective and efficient communication subsets, the development of resource sharing software, and the increased user demands for communication, economical sharing of resources, and productivity. This article is concerned with the analysis of distributed program reliability on a ring-distributed computing system. The distributed program reliability is a useful measure for reliability evaluation of distributed computing system. The distributed program reliability analyses also give a good index for designing a high-reliability-performance-distributed computing system.

Keywords: Distributed program reliability; Minimal file spanning tree; Algorithm; Ring of tree

* Corresponding author.
E-mail addresses: mlin@email.au.edu.tw (M.-S. Lin), changsam@ms5.hinet.net (M.-S. Chang), djchen@csie.nctu.edu.tw (D.-J. Chen).

PII: S0305-0548(99)00151-3

1. Introduction

Distributed computing system (DCS) has become very popular for its fault-tolerance, potential for parallel processing, and better reliability performance. One of the important issues in the design of the DCS is the reliability performance. Distributed program reliability is address to obtain this reliability measure [1–4].

An efficient network topology is quite important for the distributed computing system. The ring topology is a popular one used in high-speed network. It has been considered for IEEE 802.5 token ring, for the fiber-distributed data interface (FDDI) token ring, for the synchronous optical network (SONET), and for asynchronous transfer mode (ATM) networks. The ring network has widely used in current distributed system design.

In a *ring of tree* topology, a ring is used to connect each tree topology in the network. This architecture can be used in FDDI that consists of (1) a tree of wiring concentrators and terminal stations, and (2) a counter-rotating dual ring [5].

A large amount of work has been devoted to developing algorithms to compute measures of reliability for a DCS. One typical reliability measure for a DCS is the K-terminal reliability (KTR) [6–8]. KTR is the probability that a specified set of nodes K, which is subset of all the nodes in a DCS, remains connected in a DCS whose edges may fail independent of each other, with known probabilities. However, the KTR measure is not applicable to a practical DCS since a reliability measure for a DCS should capture the effects of redundant distribution of programs and data files. In Prasanna Kumar et al., Hari and Raghavendra and Kumar et al. [1–4], distributed program reliability (DPR) was introduced to accurately model the reliability of a DCS. For successful execution of a distributed program, it is essential that the node containing the program, other nodes that have required data files, and the edges between them be operational. DPR is thus defined as the probability that a program with distributed files can run successfully in spite of some faults occurring in the edges. In reality, the DPR problem is a logical OR-ing of Prob$\{K$-terminals are connected$\}$, but computing the conditional probabilities required could be rather nasty.

In this paper, we propose a polynomial-time algorithm to analyze the DPR of ring topology and show that solving the DPR problem on a ring of tree topology is NP-hard.

2. Notation and definitions

Notation

$D = (V, E, F)$	an undirected distributed computing system (DCS) graph with vertex set V, edge set E and data file set F.
FA_i	set of files available at node i. (Note: $F = \cup FA_i$)
p_i	reliability of edge i
q_i	$1 - p_i$
H	subset of files of F, i.e., $H \subseteq F$, and H contains the programs to be executed and all needed data files for the execution of these programs
$R(D_H)$	the DPR of D with a set H of needed files: Pr$\{$all data files in H can be accessed successfully by the executed programs in $H\}$.

Definition. A *file spanning tree* (FST) is a tree whose nodes hold all needed files in H.

Definition. A *minimal file spanning tree* (MFST) is an FST such that there exists no other FST that is a subset of it.

Definition. *Distributed program reliability* (DPR) is defined as the probability that a distributed program runs on multiple processing elements (PEs) and needs to communicate with other PEs for remote files will be executed successfully.

By the definition of MFST, the DPR can be written as

$$R(D_H) = \text{Prob(at least one MFST is operational)}$$

or

$$R(D_H) = \text{Prob}\left(\bigcup_{j=1}^{\#\text{mfst}} \text{MFST}_j\right),$$

where $\#$mfst is the number of MFSTs for a given needed file set H.

3. Computing DPR over a DCS with a ring topology

Now, we consider a DCS with a linear structure $D = (V, E, F)$ with $|E| = n$ edges in which an alternation sequence of distinct nodes and edges $(v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n)$ is given. For $1 \leqslant i \leqslant n$, let

I_i the FST which starts at edge e_i and has the minimal length
S_i the event that all edges in I_i are working
Q_i $\equiv \prod_{\text{all edge } j \in I_i} p_j$ be the probability that S_i occurs
E_i the event that there exists an operating event S_j between edges e_1 and e_i
g_i the number of I_j which lies between e_1 and e_i
x_i state of edge e_i; $x_i = 0$ if edge e_i fails; else $x_i = 1$
$\bar{A}$ the complement of event A.

It is easy to see that the DPR of a DCS with a linear structure D with $|E| = n$ edges, $R(D_H)$, can be stated as $\Pr(E_n)$. The following theorem provides a recursive method for computing $\Pr(E_n)$.

Theorem 1.

$$\Pr(E_n) = \Pr(E_{n-1}) + \sum_{i=g_{n-1}+1}^{g_n} [(1 - \Pr(E_{i-2}))q_{i-1}Q_i]$$

with the boundary conditions $\Pr(E_i) = 0$, $g_i = 0$, *and* $p_i = 0$ *for* $i \leqslant 0$.

Proof. See the appendix.

Before applying Theorem 1, we use the following procedure COMGQ to compute the values of g_i and Q_i, for $1 \leqslant i \leqslant n$, for a given linear DCS with $|E| = n$ edges.

Procedure COMGQ

// Given a DCS with a linear structure with the alternation sequence of distinct nodes and edges //
// $(v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n)$, //
// F: the set of files (including data files and programs) distributed in D; //
// H: the set of files that must be communicated each other through the edges in D; //
// FA_i: the set of files available at node v_i, for $0 \leqslant i \leqslant n$; and //
// p_i: the reliability of edge i, for $1 \leqslant i \leqslant n$, //
// this procedure computes the values of g_i and Q_i, for $1 \leqslant i \leqslant n$. //
// h (head) and t (tail) are two indexes moving among nodes. NF_i is the total number of file i //
// between nodes v_h and v_t. If there exists an FST between nodes v_h and v_t then *flag* = *true* //
// else *flag* = *false* //

```
begin
   for 2 ≤ i ≤ n do Q_i ← 0 repeat// initialize //
   p_0 ← Q_1 ← 1                   // initialize //
   h ← 0; t ← 1                    // initialize //
   for each file i ∈ F do          // initialize //
      if file i ∈ FA_h then NF_i ← 1
                       else NF_i ← 0
      endif
   repeat
   while t ≤ n do
      for each file i ∈ FA_t do NF_i ← NF_i + 1 repeat
      Q_{h+1} ← Q_{h+1} * p_t
      flag ← true
      while flag do
         for each file i ∈ H do // check if there exists an FST between //
            if NF_i = 0 then flag ← false endif //nodes v_h and v_t //
         repeat
         if flag then
            for each file i ∈ FA_h do NF_i ← NF_i − 1 repeat
            h ← h + 1
            Q_{h+1} ← Q_h/p_h
         endif
      repeat
      gt → h
      t ← t + 1
   repeat
   for 1 ≤ i ≤ n do output(g_i, Q_i) repeat
end COMGQ
```

Now, using the procedure COMGQ and Theorem 1, we are able to provide an algorithm for computing the reliability of a DCS with a linear structure.

Algorithm Reliability_Linear_DCS(D)
// Given a DCS with a linear structure $D = (V, E, F)$ with $|E| = n$ and a specified set of files H, //
// this algorithm returns the DPR of D //
Step 1: Call COMGQ to compute the values of g_i and Q_i, $1 \leqslant i \leqslant n$.
Step 2: Evaluate $\Pr(E_n)$, recursively using Theorem 1.
Step 3: Return ($\Pr(E_n)$).
end Reliability_Linear_DCS

For step 1, the computational complexity of the procedure COMGQ is $O(|E||F|)$, where $|E| = n$ and $|F| \geqslant \max((\max_{i=0}^{n}(FA_i), H))$ since the value of h in the inner while_loop is monotonously increasing and does not exceed the value of t that is the index of the outer while_loop. For step 2, by Theorem 1, $\Pr(E_i)$ can be computed in $O(g_i - g_{i-1} + 1)$. Since there are n such $\Pr(E_i)$'s to compute, we need another $O(\sum_{i=1}^{n}(g_i - g_{i-1} + 1)) = O(n + g_n - g_0) = O(n) = O(|E|)$. Therefore algorithm Reliability_Linear_DCS takes $O(|E||F|) + O(|E|) = O(|E||F|)$ time to compute the reliability of a DCS with a linear structure system.

Example 1. Consider a possible DCS of a banking systm [4,15] shown in Fig. 1. Each local disk stores some of the following information:

consumer accounts file (CAF),
administrative aids file (ADF), and
interest and exchange rates file (IXF).

Management report generation (MRG) in computers B and E indicates a query (program) to be executed for report generation. Fig. 2 shows the graph model for this system. A node represents any computer location and the links show the communication network. We assume that the query MRG(f_4) requires data CAF(f_1), ADF(f_2) and IXF(f_3) to complete its execution. Let $V = \{v_0, v_1, v_2, v_3, v_4, v_5\}$, $E = \{e_1, e_2, e_3, e_4, e_5\}$, $F = \{f_1, f_2, f_3, f_4\}$ and $H = \{f_1, f_2, f_3, f_4\}$. Applying the algorithm Reliability_Linear_DCS, we get
Step 1:

$$
\begin{aligned}
&g_0 = 0, && \text{// boundary condition //}\\
&g_1 = 1, && Q_1 = p_1,\\
&g_2 = 1, && Q_2 = p_2 p_3 p_4,\\
&g_3 = 1, && Q_3 = p_3 p_4,\\
&g_4 = 3, && Q_4 = p_4 p_5,\\
&g_5 = 4 \text{ and } && Q_5 = 0. \quad \text{// } I_5 \text{ does not exist //}
\end{aligned}
$$

Step 2:

$$\Pr(E_1) = \Pr(E_2) = \Pr(E_3) = q_0 Q_1 = p_1$$

$$
\begin{aligned}
\Pr(E_4) &= \Pr(E_3) + (1 - \Pr(E_0))q_1 Q_2 + (1 - \Pr(E_1))q_2 Q_3\\
&= p_1 + q_1 p_2 p_3 p_4 + q_1 q_2 p_3 p_4
\end{aligned}
$$

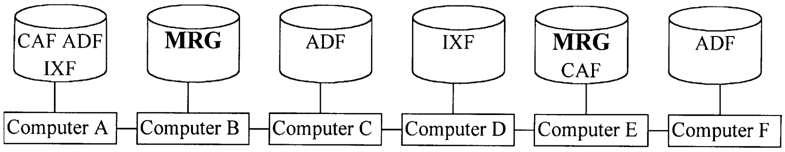

Fig. 1. A distributed banking system.

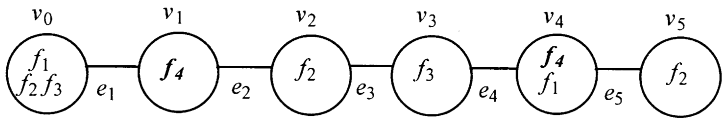

Fig. 2. The graph model for the distributed banking system in Fig. 1.

$$\Pr(E_5) = \Pr(E_4) + (1 - \Pr(E_2))q_3 Q_4$$

$$= p_1 + q_1 p_2 p_3 p_4 + q_1 q_2 p_3 p_4 + q_1 q_3 p_4 p_5.$$

A ring DCS is a DCS with a circular communication link. Each node connects two conjoining edges with two neighboring nodes. Suppose $D = (V, E, F)$ be a DCS with a ring topology. By factoring theorem [9], the DPR of D can be given as

$$R(D_H) = p_e R((D + e)_H) + q_e R((D - e)_H), \tag{1}$$

where e is an arbitrary edge of D, p_e is the reliability of edge e, $q_e \equiv 1 - p_e$, $D + e$ is the DCS D with edge $e = (u, v)$ contracted so that nodes u and v are merged into a single node and this new merged node contains all data files that previously were in nodes u and v, and $D - e$ is the DCS D with edge e deleted.

Since $D - e$ is a DCS with a linear structure with $|E| - 1$ edges, its DPR reliability can be computed by the algorithm Reliability_Linear_DCS in $O(|E||F|)$ time. Note that $D + e$ remains a DCS with a ring structure with $|E| - 1$ edges. We then apply the same analysis to $D + e$. Recursively applying Eq.1, the ring DCS D with $|E|$ edges can be decomposed into, in worst case, $|E|$ linear DCSs. So, we have an $O(|E|^2|F|)$ time algorithm for computing the reliability of a DCS with a ring structure.

Algorithm Reliability_Ring_DCS(D)
// Given a DCS with a ring structure $D = (V, E, F)$ and a specified set of files H, //
// this algorithm returns the DPR of D //

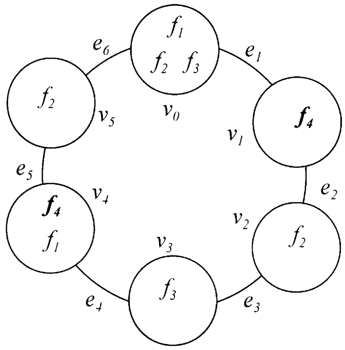

Fig. 3. A DCS with a ring structure.

Step 1: If there exists one node in V which holds all data files in H then return (1).
Step 2: Select an arbitrary edge e of D.
Step 3: $R_l \leftarrow$ Reliability_Linear_DCS($D - e$).
Step 4: $R_r \leftarrow$ Reliability_Ring_DCS($D + e$).
Step 5: Return($p_e * R_r^+ q_e * R_l$).
end Reliability_Ring_DCS

Example 2. Consider the DCS with a ring topology shown in Fig. 3. This is the DCS shown in Fig. 2 with one edge e_6 added between nodes v_5 and v_0.

Applying algorithm Reliability_Ring_DCS, we have

$$R(D_H) = q_6 R((D - e_6)_H) + p_6 R((D + e_6)_H)$$
$$= q_6 R((D - e_6)_H) + p_6\{q_5 R((D + e_6 - e_5)_H) + p_5 R((D + e_6 + e_5)_H)\}.$$

Since there exists one node in $D + e_6 + e_5$ that holds all files in H, we have $R((D + e_6 + e_5)_H) = 1$. From Example 1, it is easy to see that $R((D - e_6)_H) = \Pr(E_5)$ and $R((D + e_6 - e_5)_H) = \Pr(E_4)$. So we have

$$R(D_H) = q_6(p_1 + q_1 p_2 p_3 p_4 + q_1 q_2 p_3 p_4 + q_1 q_3 p_4 p_5)$$
$$+ p_6[q_5(p_1 + q_1 p_2 p_3 p_4 + q_1 q_2 p_3 p_4) + p_5].$$

4. Computational complexity of the DPR problem on a ring of tree topology

Complexity results are obtained by transforming known NP-hard problems to our reliability problems [10–14]. For this reason, we first state some known NP-hard problems as follows.

(i) *K- terminal reliability (KTR)*

Input: an undirected graph $G = (V, E)$ where V is the set of nodes and E is the set of edges that fail s-independent of each other with known probabilities. A set $K \subseteq V$ is distinguished with $|K| \geqslant 2$.

Output: $R(G_K)$, the probability that the set K of nodes of G is connected in G.

(ii) *Number of edge covers (#EC)*

Input: an undirected graph $G = (V, E)$.

Output: the number of edge covers for G

$\equiv |\{L \subseteq E$: each node of G is an end of some edge in $L\}|$.

(iii) *Number of vertex covers (#VC)*

Input: an undirected graph $G = (V, E)$.

Output: the number of vertex covers for G

$\equiv |\{K \subseteq V$: every edge of G has at least one end in $K\}|$.

Theorem 2. *Computing DPR for a DCS with a star topology even with each* $|FA_i| = 2$ *is* NP-*hard.*

Proof. We reduce the #EC problem to our problem. For a given network $G = (V_1, E_1)$ where $E_1 = \{e_1, e_2, \dots, e_n\}$, we construct a DCS $D = (V_2, E_2, F)$ with a star topology where $V_2 = \{s, v_1, v_2, \dots, v_n\}$, $E_2 = \{(s, v_i) \mid 1 \leqslant i \leqslant n\}$, and $F = \{f_i \mid \text{for each node } i \in G\}$. Let $FA_{v_i} = \{f_u, f_v \mid \text{if } e_i = (u, v) \in G\}$ for $1 \leqslant i \leqslant n$, $FA_s = \phi$ and $H = F$. From the construction of D, it is easy to show that there is one-to-one correspondence between one of the sets of edge covers and one FST. The DPR of D, $R(D_H)$, can be expressed as

$$R(D_H) = \sum_{\substack{\text{for all FST} \\ t \in D}} \left\{ \prod_{\substack{\text{for each} \\ \text{edge } i \in t}} p_i \prod_{\substack{\text{for each} \\ \text{edge } i \notin t}} (1 - p_i) \right\}.$$

Thus, a polynomial-time algorithm for computing $R(D_H)$ over a DCS with a star topology and each $|FA_i| = 2$ would imply an efficient algorithm for #EC problem. Since #EC problem is NP-hard, Theorem 2 follows. □

Theorem 3. *Computing DPR for a DCS with a star topology even when there are only two copies of each file is* NP-*hard.*

Proof. We reduce the #VC problem to our problem. For a given $G = (V_1, E_1)$ where $|E_1| = n$ and $V_1 = \{v_1, v_2, \dots, v_m\}$, we construct a DCS $D = (V_2, E_2, F)$ with a star topology where $V_2 = V_1 \cup \{s\}$, $E_2 = \{e_i = (s, v_i) \mid 1 \leqslant i \leqslant m\}$, and $F = \{f_i \mid \text{for all edge } i \in G\}$. Let $FA_i = \{f_j \mid \text{for all edge } j \text{ that are incident on } v_i \in G\}$ and $H = F$. From the construction of D, it is easy to show that there are only two copies of each file in D and one-to-one correspondence between one of sets of vertex covers and one FST of D. The DPR of D, $R(D_H)$, can be expressed as

$$R(D_H) = \sum_{\substack{\text{for all FST} \\ t \in D}} \left\{ \prod_{\substack{\text{for each} \\ \text{edge } i \in t}} p_i \prod_{\substack{\text{for each} \\ \text{edge } i \notin t}} (1 - p_i) \right\}$$

Since #VC problem is NP-hard, Theorem 3 follows. □

Theorem 4. *Computing DPR for a DCS with a tree topology is* NP-*hard.*

Proof. By Theorems 2 and 3, we can see that DPR problem for a DCS with a star topology, in general, is NP-hard. This implies DPR problem for a DCS with a tree topology, in general, is also NP-hard, since a DCS with a star topology is just a DCS with a tree topology that has one level branch. □

Now, We use the results of Theorems 2–4 to prove the DPR problem on ring of tree topology is NP-hard.

Theorem 5. *Computing DPR for a DCS with a ring of trees topology even with one level of tree is* NP-*hard.*

Proof. Give a DCS graph $D = (V, E, H)$ where $V = \{s, v_1, v_2, \ldots, v_n\}$ and $E = \{(s, v_i) \mid 1 \leqslant i \leqslant n\}$ with a star topology. We construct a DCS graph $D' = (V', E', H)$ from graph D, where

$$V' = \{v_1, v_2, \ldots, v_n\} \cup \{(s_j \mid 1 \leqslant j \leqslant n\}$$

and

$$E' = \{(s_j, s_{j+1}) \mid 1 \leqslant j < n\} \cup \{(s_n, s_1)\} \cup \{(s_j, v_j) \mid 1 \leqslant j \leqslant n\}.$$

It is easy to see that D' is a ring of tree topology with one level of tree. If we assume all added edges, $\{(s_j, s_{j+1}) \mid 1 \leqslant j < n\} \cup \{(s_n, s_1)\}$, of D' be perfect reliability, then we have $R(D_H) = R(D'_H)$ for any given $H \subseteq H$. By Theorems 2 and 3, computing DPR over a DCS with a star topology is NP-hard, thus, computing DPR over a DCS with a ring of tree topology with one level of tree is also NP-hard.

Theorem 6. *Computing DPR for a DCS with a ring of tree topology, in general,* NP-*hard.*

Proof. By Theorem 5, we can see that DPR problem for a DCS with a ring of tree topology even with one level of tree is NP-hard. With the same approach stated in Theorem 5, we construct a ring of tree topology with a tree topology. By Theorem 4, computing DPR over a DCS with a tree topology is NP-hard, thus, computing DPR over a DCS with a ring of tree topology is also NP-hard. □

5. Conclusions

In this paper, we investigated the problem of distributed program reliability on ring distributed computing systems. We propose a polynomial-time algorithm for computing the DPR on a ring topology. We also propose Theorems 5 and 6 to show that solving the DPR problem on a ring of trees topology is NP-hard.

Appendix

The detailed proof of Theorem 1 is as follows.

Theorem 1.

$$\Pr(E_n) = \Pr(E_{n-1}) + \sum_{i=g_{n-1}+1}^{g_n} [(1-\Pr(E_{i-2}))q_{i-1}Q_i]$$

with the boundary conditions $\Pr(E_i) = 0$, $g_i = 0$, *and* $p_i = 0$, *for* $i \leqslant 0$.

Proof.

$$\Pr(E_n) = \Pr\left(E_{n-1} \bigcup_{i=g_{n-1}+1}^{g_n} S_i\right) = \Pr(E_{n-1}) + \Pr\left(\overline{E_{n-1}} \cap \left(\bigcup_{i=g_{n-1}+1}^{g_n} S_i\right)\right). \tag{A.1}$$

For the term $\Pr(\overline{E_{n-1}} \cap (\bigcup_{i=g_{n-1}+1}^{g_n} S_i))$ in Eq. (1), we have

$$\begin{aligned}\Pr\left(\overline{E_{n-1}} \cap \left(\bigcup_{i=g_{n-1}+1}^{g_n} S_i\right)\right) &= \Pr(\overline{E_{n-1}} \cap S_{g_{n-1}+1}) + \Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+1}} \cap S_{g_{n-1}+2}) \\ &+ \Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+1}} \cap \overline{S_{g_{n-1}+2}} \cap S_{g_{n-1}+3} \\ &+ \Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+1}} \cap \cdots \cap \overline{S_{g_n-1}} \cap S_{g_n}).\end{aligned} \tag{A.2}$$

Since $S_i = S_{i+1} \cap \{x_i = 1\}$ for $g_{n-1}+1 \leqslant i \leqslant g_n$, we have $S_i \subset S_{i+1}$ and $\overline{S_i} \cap \overline{S}_{i+1} = \overline{S_{i+1}}$.

$$\begin{aligned}\Pr\left(\overline{E_{n-1}} \cap \left(\bigcup_{i=g_{n-1}+1}^{g_n} S_i\right)\right) &= \Pr(\overline{E_{n-1}} \cap S_{g_{n-1}+1}) + \Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+1}} \cap S_{g_{n-1}+2}) \\ &+ \Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+2}} \cap S_{g_{n-1}+3}) + \cdots + \Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}}} \cap S_{g_n}) \\ &= \Pr(\overline{E_{n-1}} \cap S_{g_{n-1}+1}) + \sum_{i=g_{n-1}+2}^{g_n} \Pr(\overline{E_{n-1}} \cap \overline{S_{i-1}} \cap S_i) \\ &= \Pr(\overline{E_{g_{n-1}-1}} \cap \{x_{g_{n-1}} = 0\} \cap S_{g_{n-1}+1}) + \sum_{i=g_{n-1}+2}^{g_n} \Pr(\overline{E_{i-2}} \cap \{x_{i-1} = 0\} \cap S_i) \\ &= \sum_{i=g_{n-1}+1}^{g_n} \Pr(\overline{E_{i-2}} \cap \{x_{i-1} = 0\} \cap S_i).\end{aligned}$$

Note that the events E_{i-2}, $\{x_{i-1} = 0\}$, and S_i are disjoint with each other. We have

$$\Pr(\overline{E_{i-2}} \cap \{x_{i-1} = 0\} \cap S_i) = \Pr(\overline{E_{i-2}})\Pr(\{x_{i-1} = 0\})\Pr(S_i).$$

So

$$\begin{aligned}\Pr\left(\overline{E_{n-1}} \cap \left(\bigcup_{i=g_{n-1}+1}^{g_n} S_i\right)\right) &= \sum_{i=g_{n-1}+1}^{g_n} \Pr(\overline{E_{i-2}})\Pr(\{x_{i-1} = 0\})\Pr(S_i) \\ &= \sum_{i=g_{n-1}+1}^{g_n} [(1-\Pr(E_{i-2}))q_{i-1}Q_i]\end{aligned} \tag{A.3}$$

By Eqs. (1) and (3), we obtain Theorem 1. □

References

[1] Prasanna Kumar VK, Hariri S, Raghavendra CS. Distributed program reliability analysis. IEEE Transaction Software Engineering 1986;SE-12:42–50.
[2] Hariri S, Raghavendra CS. SYREL: A symbolic reliability algorithm based on path and cutset methods. USC Tech. Rep. 1984.
[3] Kumar A, Rai S, Agrawal DP. Reliability evaluation algorithms for distributed systems. In Proceedings IEEE INFOCOM 88, 1988. p. 851–60.
[4] Kumar A, Rai S, Agrawal DP. On computer communication network reliability under task execution constraints. IEEE Journal on Selected Areas in Communication 1988;6(8):1393–9.
[5] Jain R. FDDI handbook: high speed networking using fiber and other media. New York: Addison-Wesley, 1994.
[6] Jiahnsheng Yin, Charles Silio Jr. B. K-terminal reliability in ring networks. IEEE Transactions on Reliability 1994;43(3):389–400.
[7] Logothetis D, Trivedi KS. Reliability analysis of the double counter-rotating ring with concentrator attachment. IEEE Transactions On Networking 1994;2(5):520–32.
[8] Chen DJ, Chang MS, Yang CL, Ku Kuo-Lung. Multimedia task reliability analysis based on token ring network. 1996 International Conference on Parallel and Distributed System, 1996. p. 265–72.
[9] Wood RK. Factoring algorithms for computing K-terminal network reliability. IEEE Transaction Reliability 1986;R-35:269–78.
[10] Rosenthal A. A computer scientist looks at reliability computations. In: Reliability and fault tree analysis SLAM. 1975. p. 133–52.
[11] Valiant LG. The complexity of enumeration and reliability problems. SIAM Journal of Computing 1979;8:410–21.
[12] Ball MO, Provan JS, Shier DR. Reliability covering problems. Networks 1991;21:345–57.
[13] Provan JS, Ball MO. The complexity of counting cuts and of computing the probability that a graph is connected. SIAM Journal of Computing 1983;12(4):777–88.
[14] Provan JS. The complexity of reliability computations in planar and acyclic graphs. SIAM Journal on Computing 1986;15:694–702.
[15] Sheppard DA. Standard for banking communication system. IEEE Transaction Computer 1987;92–5.

Min-Sheng Lin received his MS & Ph.D. in Computer Science & Information Engineering from National Chiao Tung University (Hsin Chu, Taiwan). He is currently an associate professor at Aletheia University (Taipei, Taiwan). His research interests include reliability and performance evaluation of distributed computing systems.

Ming-Sang Chang received BS degree in Electronic Engineering from National Taiwan University of Science and Technology (Taipei, Taiwan), MS degree in information engineering from TamKang University (Taipei, Taiwan), and Ph.D. degree in Computer Science & Information Engineering from National Chiao Tung University (Hsin Chu, Taiwan). He is currently working in Chunghwa Telecommunication Training Institute (Taipei, Taiwan). His research interests include computer network, performance evaluation, distributed system, and reliability evaluation.

Deng-Jyi Chen received the BS degree in Computer Science from Missouri State University (Cape Girardeau), and MS and Ph.D. degree in Computer Science from the University of Texas (Arlington). He is now a professor at National Chiao Tung University (Hsin Chu, Taiwan). His papers have been published in more than 100 journals and conference papers in the area of reliability and performance modeling of distributed systems, computer networks, object-oriented systems, and software reuse. Professor Chen works very closely with industrial sectors and provides consulting for many local companies (both for software and hardware companies). So far, he has been a chief leader of designing and implementing two commercial products which are now marketing around the world.

Kuo-Lung Ku received his bachelor and master degree from Chiao Tung University, and his Ph.D. in Electrical Engineering from the University of Washington. His major research areas include computer architecture, real-time computation and security. He is working in Chung Shan Institute of Science and Technology.