The Distributed Program Reliability Analysis on Ring-Type Topologies

環狀拓樸邏輯分散式程式可靠度分析

Ming-Sang Chang [1]
Department of Information Management
Lunghwa University of Science and Technology
300,Wan Shou Rd., Sec. 1, Kueishan Taoyuan, Taiwan, R.O.C.
Email:changsam@ms5.hinet.net

Min-Sheng Lin
Department of Electric Engineering National Taipei University of Technology 1, Sec. 3, Chung Hsiao E. Rd., Taipei, Taiwan, R.O.C
Email:mslin@ee.ntut.edu.tw

Deng-Jyi Chen
Institute of Computer Science and Information Engineering National Chiao Tung University 1001 Ta Hsueh Road, Hsin Chu, Taiwan, R.O.C.
Email: djchen@csie.nctu.edu.tw

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 topologies and show that solving the DPR problem on a ring of trees topology is *NP*-hard.

Keywords: Distributed Program Reliability, Minimal File Spanning Tree, Algorithm, Ring of Tree

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 independently 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 [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 the 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

[1] All correspondences should be sent to **MING-SANG CHANG (changsam@ms5.hinet.net)**

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 a 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

$R\,(D_H)$=Prob(at least one MFST is operational), or

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

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 $1\le i\le n$, 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
$\overline{A}$	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 $i \le 0$.

Proof. See the appendix.

Before applying the Theorem 1, we use the following procedure COMGQ to compute the values of g_i and Q_i, for $1 \le i \le n$, 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 $0\le i \le n$; and //

// p_i: the reliability of edge i, for $1\le i \le n$, //

```
// this procedure computes the values of $g_i$ and $Q_i$, for $1 \le i \le n$.//
//$h$ (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 a FST between nodes $v_h$ and $v_t$ then $flag=true$ //
// else $flag= false$ //
begin
    for $2 \le i \le n$ do $Q_i \leftarrow 0$ repeat // initialize //
    $p_0 \leftarrow Q_1 \leftarrow 1$                // initialize //
    $h \leftarrow 0$; $t \leftarrow 1$               // initialize //
    for each file $i \in F$ do                       // initialize //
        if file $i \in FA_h$ then $NF_i \leftarrow 1$
            else $NF_i \leftarrow 0$
        endif
    repeat
    while $t \le n$ do
        for each file $i \in FA_t$ do $NF_i \leftarrow NF_i+1$ repeat
        $Q_{h+1} \leftarrow Q_{h+1} * p_t$
        $flag \leftarrow true$
        while $flag$ do
            for each file $i \in H$ do        // check if there exists a FST between //
                if $NF_i=0$ then $flag \leftarrow false$ endif //nodes $v_h$ and $v_t$ //
            repeat
            if $flag$ then
        for each file $i \in FA_h$ do $NF_i \leftarrow NF_i-1$ repeat
                $h \leftarrow h+1$
                $Q_{h+1} \leftarrow Q_h / p_h$
            endif
        repeat
            $g_t \leftarrow h$
            $t \leftarrow t+1$
    repeat
    for $1 \le i \le n$ 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 $Q_i$, $1 \le i \le n$.
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 $|F| \ge max((max_{i=0}^{n}(FA_i), H))$ since the value of h in the inner while_loop is monotonously increasing and doesn't 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:

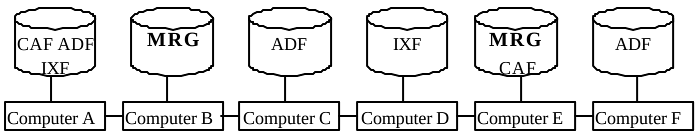


Figure 1. A distributed banking system

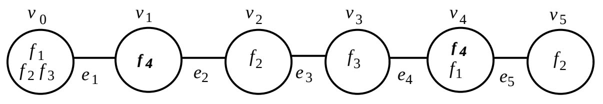


Figure 2. The graph model for the distributed banking system in figure 1.

Consider a possible DCS of a banking system [4,17] shown in figure 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. Figure 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:

$g_0=0$, // boundary condition //

$g_1=1$,	$Q_1=p_1$,
$g_2=1$,	$Q_2=p_2p_3p_4$,
$g_3=1$,	$Q_3=p_3p_4$,
$g_4=3$,	$Q_4=p_4p_5$,
$g_5=4$, and	$Q_5=0$. // I_5 does not exist //

Step 2:

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

$$Pr(E_4) = Pr(E_3)+(1-Pr(E_0))q_1Q_2+(1-Pr(E_1))q_2Q_3$$

$$= p_1+q_1p_2p_3p_4+q_1q_2p_3p_4$$

$$Pr(E_5) = Pr(E_4)+(1-Pr(E_2))q_3Q_4$$

$$= p_1+q_1p_2p_3p_4+q_1q_2p_3p_4+q_1q_3p_4p_5.$$

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

$$R(D_H)=p_eR((D+e)_H)+q_eR((D-e)_H), \quad \text{(Eq. 1)}$$

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

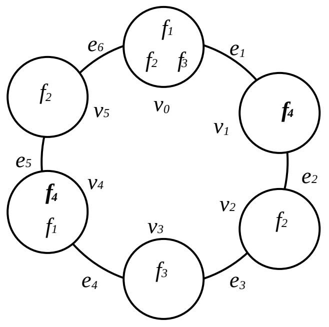
$D-e$ is the DCS D with edge e deleted.

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 Equation (Eq.1), the ring DCS D with $|E|$ edges can be decomposed into, in worst case, $|E|$ linear DCSs. So, we have a $O(|E|^2|F|)$ 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 $D$ //
Step 1: If there exists one node in $V$ holds all data files in $H$ then return (1).
Step 2. Select an arbitrary edge $e$ of $D$.
Step 3: $R_l \longleftarrow$Reliability_Linear_DCS($D-e$).
Step 4: $R_r \longleftarrow$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 Figure 3. This is the DCS shown in Figure 2 with one edge e_6 added between nodes v_5 and v_0.



Program f_4 needs data files f_1, f_2, f_3 for its execuitn.

Figure 3. A DCS with a ring structure

Applying algorithm Reliability_Ring_DCS, we have

$$R(D_H)=q_6R((D\text{-}e_6)_H) + p_6R((D+e_6)_H)$$

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

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

$$R(D_H) = q_6(p_1+q_1p_2p_3p_4+q_1q_2p_3p_4+q_1q_3p_4p_5) + p_6[q_5(p_1+q_1p_2p_3p_4+q_1q_2p_3p_4)+p_5].$$

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*-independently of each other with known probabilities. A set $K \subseteq V$ is distinguished with $|K| \geq 2$.

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

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

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 $V_1=\{v_1, v_2, ..., v_m\}$, we construct a DCS $D=(V_2, E_2, F)$ with a star topology where $V_2=V_1 \cup \{s\}$, $E_2=\{e_i=(s, v_i) \mid 1 \leq i \leq m\}$, and $F=\{f_i \mid$ for all edge $i \in G\}$. Let $FA_i=\{ f_j \mid$ for all edge j 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

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

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 Theorem 2, 3, and 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 $V=\{s, v_1, v_2, ..., v_n\}$ and $E=\{(s, v_i) \mid 1 \leq i \leq n\}$ with a star topology. We construct a DCS graph $D'=(V', E', H)$ from graph D, where

$V'=\{ v_1, v_2, ..., v_n\} \cup \{(s_j \mid 1 \leq j \leq n\}$ and

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

It is easy to see that D' is a ring of tree topology with one level of tree. If we assume all added edges, $\{(s_j, s_{j+1}) \mid 1 \leq j < n\} \cup \{(s_n, s_1)\}$, of D' be perfect reliability, then we have $R(D_H)=R(D'_H)$ for any given $H \subseteq H$. By Theorem 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, is 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 Theorem 5 and Theorem 6 to show that solving the DPR problem on a ring of trees topology is *NP*-hard.

6. 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 $i \le 0$.

Proof.

$$Pr(E_n) = Pr(E_{n-1} \bigcup_{i=g_{n-1}+1}^{g_n} S_i)$$

$$= Pr(E_{n-1}) + Pr(\overline{E_{n-1}} \cap (\bigcup_{i=g_{n-1}+1}^{g_n} S_i))$$

(Eq.2)

For the term $Pr(\overline{E_{n-1}} \cap (\bigcup_{i=g_{n-1}+1}^{g_n} S_i))$ in (Eq.2), we have

$$Pr(\overline{E_{n-1}} \cap (\bigcup_{i=g_{n-1}+1}^{g_n} S_i))$$

$$= Pr(\overline{E_{n-1}} \cap S_{g_{n-1}+1}) + Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+1}} \cap S_{g_{n-1}+2}) + Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+1}} \cap \overline{S_{g_{n-1}+2}} \cap S_{g_{n-1}+3}) + \ldots$$

$$+ Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+1}} \cap \ldots \cap \overline{S_{g_n-1}} \cap S_{g_n}).$$

(Eq.3)

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

$$Pr(\overline{E_{n-1}} \cap (\bigcup_{i=g_{n-1}+1}^{g_n} S_i))$$

$$= Pr(\overline{E_{n-1}} \cap S_{g_{n-1}+1}) + Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+1}} \cap S_{g_{n-1}+2}) + Pr(\overline{E_{n-1}} \cap \overline{S_{g_{n-1}+2}} \cap S_{g_{n-1}+3}) + \ldots$$

$$+ Pr(\overline{E_{n-1}} \cap \overline{S_{g_n-1}} \cap S_{g_n})$$

$$= \Pr(\overline{E_{n-1}} \cap S_{g_{n-1}+1}) + \sum_{i=g_{n-1}+2}^{g_n} \Pr(\overline{E_{n-1}} \cap \overline{S_{i-1}} \cap S_i)$$

$$= Pr(\overline{E_{g_{n-1}-1}} \cap \{x_{g_{n-1}}=0\} \cap S_{g_{n-1}+1})$$

$$+ \sum_{i=g_{n-1}+2}^{g_n} \Pr(\overline{E_{i-2}} \cap \{x_{i-1}=0\} \cap S_i)$$

$$= \sum_{i=g_{n-1}+1}^{g_n} \Pr(\overline{E_{i-2}} \cap \{x_{i-1}=0\} \cap S_i)$$

Note that the events $\overline{E_{i-2}}$, $\{x_{i-1}=0\}$, 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

$$Pr(\overline{E_{n-1}} \cap (\bigcup_{i=g_{n-1}+1}^{g_n} S_i))$$

$$= \sum_{i=g_{n-1}+1}^{g_n} \Pr(\overline{E_{i-2}}) * \Pr(\{x_{i-1}=0\}) * \Pr(S_i) =$$

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

(Eq.4)

By equations (Eq.2) and (Eq.4), we obtain Theorem 1.

Reference

[1] V. K. Prasanna Kumar, S. Hariri and C. S. Raghavendra, "Distributed program reliability analysis," *IEEE Trans. Software Eng.*, vol. SE-12, pp. 42-50, Jan. 1986.

[2] S. Hariri and C.S. Raghavendra, "SYREL: A Symbolic Reliability Algorithm based on Path and Cutset Methods", USC Tech. Rep., 1984.

[3] A. Kumar, S. Rai and D.P. Agrawal, "Reliability Evaluation Algorithms for Distributed Systems", in *Proc. IEEE INFOCOM* 88, pp.851-860, 1988.

[4] A. Kumar, S. Rai and D.P. Agrawal, "On Computer Communication Network Reliability Under Task Execution Constraints", *IEEE Journal on Selected Areas in Communication*, Vol.6, No.8, pp. 1393-1399, Oct.1988.

[5] Raj Jain, "FDDI Handbook : High Speed Networking Using Fiber and Other Media" , Addison-Wesley Publishing Company, 1994

[6] Jiahnsheng Yin, Charles B. Silio Jr., "K-Terminal Reliability In Ring Networks," IEEE Trans. on Reliability, vol. 43, no. 3, pp. 389-400,1994.

[7] Dimitris Logothetis, Kishor S. Trivedi, " Reliability Analysis of the Double Counter-rotating Ring with Concentrator Attachment," IEEE Trans. On Networking, vol.2, no.5, pp. 520-532,1994.

[8] D.J.Chen, M.S.Chang, C.L.Yang, Kuo-Lung Ku," Multimedia Task Reliability Analysis Based on Token Ring Network," 1996 Int. Conference on Parallel and Distributed System, pp.265-272,1996.

[9] R. Kevin Wood, "Factoring Algorithms for Computing K-terminal Network Reliability", *IEEE Trans. Reliability*, Vol. R-35, pp.269-278, Aug. 1986.

[10] A. Rosenthal, "A Computer Scientist Looks at Reliability Computations, "*in: Reliability and Fault tree Analysis SLAM*, 1975, pp. 133-152.

[11] L. G. Valiant, "The Complexity of Enumeration and Reliability Problems," *SIAM J. Computing*, vol. 8, pp. 410-421, 1979.

[12] M. O. Ball , J. S. Provan and D. R. Shier, "Reliability Covering Problems," *Networks*, vol. 21, pp. 345-357, 1991

[13] J. S. Provan and M. O. Ball, "The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected," *SIAM J. Computing*, vol. 12, no. 4, pp. 777-788, Nov. 1983.

[14] J. S. Provan, "The complexity of reliability computations in planar and acyclic graphs", *SIAM Journal on Computing 15* (1986) 694-702.

[15] M.S. Lin, "Program Reliability Analysis in Distributed Computing System", Ph.D. Dissertation, 1994; National Chiao Tung University, Taiwan.

[16] Min-Sheng Lin, Deng-Jyi Chen, "The Computational Complexity of the Reliability Problem on Distributed Systems", Information Processing Letters 64 (1997) 143-147.

[17] D.A. Sheppard, "Standard for banking communication system", IEEE Trans. Computer, 1987 Nov, pp 92-95.