1998 International Computer Symposium
Workshop on Computer Networks, Internet, and Multimedia
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

THE DISTRIBUTED PROGRAM RELIABILITY ANALYSIS IN RING NETWORKS

Ming-Sang Chang, Deng-Jyi Chen
Institute of Computer Science and Information Engineering
National Chiao Tung University, Hsin Chu, Taiwan, R.O.C.
Email: {djchen,mschang}@csie.nctu.edu.tw

Min-Sheng Lin
Department of Information Management
Tamsui Oxford University College, Tamsui, Taipei, Taiwan, R.O.C.

Kuo-Lung Ku
Chung-Shan Institute of Science and Technology
Tao-Yuan, Taiwan, R.O.C.

ABSTRACT

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 this paper we are interested in DCS with ring topologies. We propose polynomial-time algorithms to analyze the distributed program reliability of dual ring topology and show that solving the DPR on a ring of trees topology is NP-hard.

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.

An efficient network topology is quite important for the distributed computing system. The ring topology is a popular one used in high speed network. The counter rotating dual ring is the most popular topology of ring that has been considered for IEEE 802.5 token ring, subsequently for the fiber distributed data interface (FDDI) token ring, and recently for the synchronous optical network (SONET), and for asynchronous transfer mode (ATM) networks [1]. Thus, a distributed computing system design in ring network will be a common choice.

In ring network, we will consider two topologies. One is counter rotating dual ring and the other is ring of tree topology. The topology consists of several trees hanging from a ring. This is the so-called ring of tree topology. The ring of tree topology consists of (1) a tree of wiring concentrators and terminal stations, and (2) a counter-rotating dual ring [2,3].

In a distributed computing system (DCS), a program usually needs data files on other stations in order to complete its mission. There are two reliability cases can be discussed in a distributed computing system based on the way of data files distribution.

1) No replicated data files and programs: in this case, we consider that only one copy of each data file or program is allowed in the DCS.
2) Replicated data files and programs: in this case, we consider that two or more same data files and programs can exist in the DCS.

In this paper we propose polynomial-time algorithms to analyze the DPR of dual ring topology and show that solving the DPR on a ring of tree topology is NP-hard. In section 2 the definitions and notation are given for this paper. In section 3 we focus on ring network with no replicated data files and programs. The DPR analysis on ring networks with replicated data files and programs is in section 4. Finally, summary and concluding remarks are given.

2. DEFINITIONS AND NOTATION

2.1 Definitions

FST	Files Spanning Tree (FST) is a spanning tree that connects a station that runs the program under consideration to other stations such that this spanning tree holds all the files needed for the program under consideration
MFST	Minimal Files Spanning Tree (MFST) is a FST such that there exists no other FST which is subset of it

2.2 Notation

C-R	Counter-Rotating ring network
k	the number of files that a program needs from other nodes
$R^{k}_{C-R}(t)$	the reliability of C-R for k files at time t
i_j	The node number, from clockwise orientation based on the receiving node, that contains the files required for the computed program.
q	the probability of failure for link and node
p	1- q

$Sg(x_i, x_j)$	No failed links and nodes from node x_i to x_j clockwise exclude node x_i and x_j in the ring network
$Sg[x_i, x_j]$	No failed links and nodes from node x_i to x_j clockwise in the ring network
$D=(V, E, F)$	an undirected Distributed Computing System (DCS) graph with vertex set V, edge set E and data files F.
FA_i	set of files available at node i.
H	subset of files of F, i.e., $H \subseteq$ F, and H contains the programs to be executed and all needed files for the execution of these programs
$R(D_H)$	the DPR of D with a set H of needed files: Pr{all files in H can be accessed successfully by the executed programs in H}.

3. THE DISTRIBUTED PROGRAM RELIABILITY ANALYSIS WITH NO REPLICATED DATA FILES AND PROGRAMS

In a distributed computing system (DCS), a program usually needs data files on other stations in order to complete its mission. In [4], Kumar gives the definition of Distributed Program Reliability (DPR) and formulates the DPR as $DPR = \Pr\{\bigcup_i MFST_i\}$

Where *MFST* is a Minimal Files Spanning Trees.

It should be noted that reliability algorithms presented in [4-7] for the DPR analysis are for general network topologies and the complexity of these algorithms is very high, in general, it has been proved to be NP-hard [8-9]. Futhermore, for considering data files access time in DPR evaluation, these algorithms can not be used directly. Also, these reliability algorithms are not designed specifically for ring networks. Thus, in the following sections, we propose efficient polynomial time reliability algorithms for DPR analysis under ring networks. There are two reliability cases can be discussed in a distributed computing system based on the way of their data files distribution.

1) No replicated data files and programs: in this case, we consider that only one copy of each data file or program is allowed in the DCS.
2) Replicated data files and programs: in this case, we consider that two or more same data files and programs can exist in the DCS.

3.1 No Replicated Data files and No Replicated Programs

In this case, we consider that only one copy of each data file or program is allowed in the DCS. We use three implementation of ring network to analyze the reliability. The three topologies are 1) Counter-Rotating dual ring, 2) a tree of concentrator, 3) Counter-Rotating dual ring of concentrator trees.

3.2 Counter-Rotating Dual Ring (C-R Dual Ring)

In discussing the reliability derivation, we first number the receiving node as N and other nodes from 1 to N-1 clockwise as shown in Figure 1.

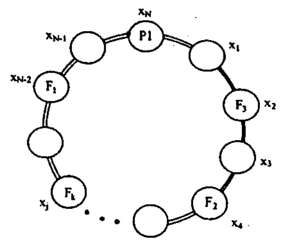


Figure 1. An example of k data files needed for N-station ring network

Let the total number of nodes which contain the data files for some program is k. In other words, there are k number of nodes that contain data files required for some programs in the C-R network. Suppose that program P_1 needs data files F_1, F_2, F_3, and F_k, then the sending nodes are x_2, x_4, x_j and x_{N-2} ; the receiving node is x_N. In a clockwise direction from node x_N, data file F_3 in node x_2 is the first needed data file. So i_1=2. Similarly, data file F_2 in node x_4 is the second needed data file. Thus, i_2=4. Likewise F_1 in node x_{N-2} is the last needed data file, and i_k =N-2.

Lemma 1: When there are no failed links and no failed nodes from node x_{i_1} to x_N clockwise in the C-R network and assumming that each node and link has reliability P_x and P_e respectively, the reliability of no failed links and no failed nodes from node x_{i_1} to x_N clockwise in the C-R network is

$\Pr\{S_g[x_{i_1}, x_N]\} = P_x^{N-i_1+1} P_e^{N-i_1}$

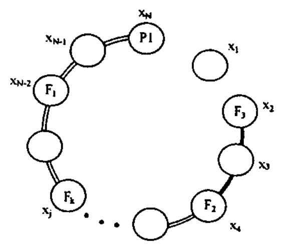


Fig.2 Snapshot 1 of Figure 1 ($S_g[x_i, x_j]$)

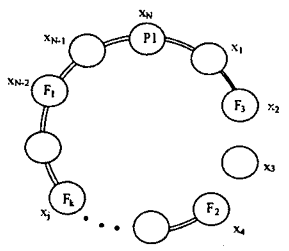


Fig.3 Snapshot 2 of Figure 1 ($S_g[x_i, x_j]$)

Proof: This lemma can easily be proved by using the Figure 2. From the Figure 2, it is clear that the first node that contains the data files required for some program is x_2, with node number i_1=2. which denoted as x_{i_1}. Based on the condition that starting from this node there are no failed nodes and links to node x_N, we can easily to see that there are N- i_1+ 1 good nodes and N- i_1 good links. Thus the reliability of no failed links and nodes from node x_{i_1} to x_N clockwise in the C-R network is

$$\Pr\{ S_g[x_{i_1}, x_N]\} = P_x^{N-i_1+1} P_e^{N-i_1}$$

Lemma 2: When there are no failed links and no failed nodes from node x_{i_2} to x_{i_1} clockwise in the ring network and assumming that each node and link has reliability p_x and p_e respectively, the reliability of no failed links and no failed nodes from node x_{i_2} to x_{i_1} clockwise in the ring network is

$$\Pr\{Sg[x_{i_2}, x_{i_1}]\} = p_x^{N-i_2+i_1+1} p_e^{N-i_2+i_1}$$

Proof: In the same manner as we discussed in the proof of lemma 1, we can easily obtain this equation using Figure 3. The only difference is that starting node is x_{i_2} with node number i_2 =4 and the terminal node is x_2 with node number i_1 =2 in this case.

Corollary 1:

$$\Pr\{Sg[x_{i_k}, x_{i_{k-1}}]\} = p_x^{N-i_k+i_{k-1}+1} p_e^{N-i_k+i_{k-1}} \quad \text{and}$$

$$\Pr\{Sg[x_N, x_{i_k}]\} = p_x^{i_k+1} p_e^{i_k}$$

Proof: These equations can be obtained for the same reasons as stated in lemma 1 and 2.

Theorem 1: The DPR for the no replicated data files and no replicated programs in the C-R network is

$R_{C-R}^k =$

$$P_x^{N-i_1+1} P_e^{N-i_1} + P_x^{N-i_2+i_1+1} P_e^{N-i_2+i_1} + \bullet\bullet\bullet + P_x^{N-i_k+i_{k-1}+1} P_e^{N-i_k+i_{k-1}} + P_x^{i_k+1} P_e^{i_k} - kP_x^N P_e^N$$

Proof: Let $\Pr\{All\}$ be the probability that all nodes and links are operational, then we have

$$\Pr\{All\} = P_x^N P_e^N$$

We have discussed various transmission cases and derived their reliability equations in lemma 1, lemma 2, and corollary 1. Based on these equations, we can easily to see that

$$R_{C-R}^k = \Pr\{ S_g[x_{i_1}, x_N] \cup S_g[x_{i_2}, x_{i_1}] \cup \bullet\bullet\bullet \cup S_g[x_{i_k}, x_{i_{k-1}}]$$
$$\cup S_g[x_N, x_{i_k}]\}$$
$$= \Pr\{ S_g[x_{i_1}, x_N] \cup (S_g[x_{i_2}, x_{i_1}] \cup \bullet\bullet\bullet \cup S_g[x_{i_k}, x_{i_{k-1}}] \cup S_g[x_N, x_{i_k}])\}$$
$$= \Pr\{ S_g[x_{i_1}, x_N]\} + \Pr\{ S_g[x_{i_2}, x_{i_1}] \cup \bullet\bullet\bullet \cup S_g[x_{i_k}, x_{i_{k-1}}]$$
$$\cup S_g[x_N, x_{i_k}]\} - \Pr\{All\}$$
$$= P_x^{N-i_1+1} P_e^{N-i_1} + \Pr\{ S_g[x_{i_2}, x_{i_1}] \cup (\bullet\bullet\bullet \cup S_g[x_{i_k}, x_{i_{k-1}}] \cup S_g[x_N, x_{i_k}])\} -$$
$$\Pr\{All\} = \bullet\bullet\bullet$$
$$= P_x^{N-i_1+1} P_e^{N-i_1} + P_x^{N-i_2+i_1+1} P_e^{N-i_2+i_1} + \bullet\bullet\bullet + P_x^{N-i_k+i_{k-1}+1} P_e^{N-i_k+i_{k-1}} + P_x^{i_k+1} P_e^{i_k} - kP_x^N P_e^N$$

Example 1: A distributed computing system with 2 unique programs and 5 unique data files distributed in a 10 processors C-R network as shown in Fig. 4.

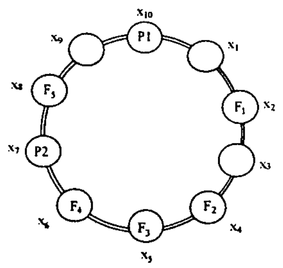


Figure 4. An example for 10 stations

In example 1, let $p_x = 1$, $p_e = 0.9$, and let P1 need data files F1, F3, and F5 to complete execution.

Let N=10, since there are ten stations in this ring network. We number these nodes as shown in Figure 4. Note that we first number the node containing P1 as $x_N = x_{10}$, the remainder from 1 to 9 in a clockwise direction.

$i_1 = 2$, since F1 is on node x_2, $i_2 = 5$, since F3 is on node x_5 and $i_3 = 8$, since F5 is on node x_8. There are three sending nodes, so k=3.

$$\Pr\{ S_g[x_{i_1}, x_{10}]\} = \Pr\{ S_g[x_2, x_{10}]\} = (1)^9 (0.9)^8 = (0.9)^8$$
$$\Pr\{ S_g[x_{i_2}, x_{i_1}]\} = \Pr\{ S_g[x_5, x_2]\} = (1)^8 (0.9)^7 = (0.9)^7$$
$$\Pr\{ S_g[x_{i_3}, x_{i_2}]\} = \Pr\{ S_g[x_8, x_5]\} = (1)^8 (0.9)^7 = (0.9)^7$$
$$\Pr\{ S_g[x_{10}, x_{i_3}]\} = \Pr\{ S_g[x_{10}, x_8]\} = (1)^9 (0.9)^8 = (0.9)^8$$
$$\Pr\{All\} = P_x^N P_e^N = (1)^{10} (0.9)^{10} = (0.9)^{10}$$

Then the reliability of P1 is computed as

$$R_{C-R}^k = \Pr\{ S_g[x_{i_1}, x_{10}] \cup S_g[x_{i_2}, x_{i_1}] \cup S_g[x_{i_3}, x_{i_2}] \cup S_g[x_{10}, x_{i_3}]\}$$
$$= (0.9)^8 + (0.9)^7 + (0.9)^7 + (0.9)^8 - 3(0.9)^{10} = 0.7714929$$

The above analysis is for a fixed distribution of data files. That is, the distribution of data files is given, and we use it to compute the DPR. Suppose that we have exponential distributed mean value for data files distribution, then the equations formulated in lemma 1, lemma 2, and corollary 1, can be written as

$$\Pr\{ S_g[x_{i_1}, x_N]\} = e^{-(N-i_1+1)\lambda_x t} \cdot e^{-(N-i_1)\lambda_e t}$$
$$\Pr\{ S_g[x_{i_2}, x_{i_1}]\} = e^{-(N-i_2+i_1+1)\lambda_x t} \cdot e^{-(N-i_2+i_1)\lambda_e t}$$

....

$$\Pr\{ S_g[x_{i_k}, x_{i_{k-1}}]\} = e^{-(N-i_k+i_{k-1}+1)\lambda_x t} \cdot e^{-(N-i_k+i_{k-1})\lambda_e t}$$

$$\Pr\{ S_g[x_N, x_{i_k}]\} = e^{-(i_k+1)\lambda_x t} \cdot e^{-i_k \lambda_e t} \quad \text{and} \quad \Pr\{All\} = P_x^N P_e^N$$

With these new equations, the reliability equation presented in Theorem 1 for no replicated data files and no replicated programs in the C-R network can be written as

$$R_{C-R}^k = \Pr\{ S_g[x_{i_1}, x_N] \cup S_g[x_{i_2}, x_{i_1}] \cup \bullet\bullet\bullet \cup S_g[x_{i_k}, x_{i_{k-1}}]$$
$$\cup S_g[x_N, x_{i_k}]\}$$

$$= \frac{1}{C_k^{N-1}} \sum_{i_1=1}^{N-k} \sum_{i_2=i_1+1}^{N-k+1} \bullet\bullet\bullet \sum_{i_k=i_{k-1}+1}^{N-1} \left(\begin{array}{l} e^{-(N-i_1+1)\lambda_x t} \cdot e^{-(N-i_1)\lambda_e t} + \\ e^{-(N-i_2+i_1+1)\lambda_x t} \cdot e^{-(N-i_2+i_1)\lambda_e t} + \\ \cdots + \\ e^{-(N-i_k+i_{k-1}+1)\lambda_x t} \cdot e^{-(N-i_k+i_{k-1})\lambda_e t} + \\ e^{-(i_k+1)\lambda_x t} \cdot e^{-i_k \lambda_e t} - \\ k e^{-N\lambda_x t} \cdot e^{-N\lambda_e t} \end{array} \right)$$

and the mean time to failure is

$$\int_0^\infty R_{C-R}^k(t)dt$$

3.3 Tree of Concentrator

In order to reduce the size of graph and therefore reduce the state space of the associated reliability problem, reliability-preserving reductions can be applied. The following reductions are designed and developed to speed up the reliability evaluation [8,15].

• *degree-1 reduction*: Degree-1 reduction removes 1)degree-1 nodes which contain none of needed data files and programs under consideration and 2)their incident edges.

• *series reduction*: Let $e_a=(u, v)$ and $e_b=(v, w)$ be two series edges in a DCS graph D such that $degree(v)$=2 and $FA_v \cap H=\varnothing$, i.e., node v contains no required data files or programs to be executed. Then a DCS graph D' is obtained by replacing e_a and e_b with a single edge $e_c=(u, w)$ such that $p_{e_c}=p_{e_a}*p_v*p_{e_b}$.

- *degree-2 reduction*: Suppose node v is a reducible node; then one can apply series reduction on node v and move data files and programs within node v to one of its adjacent nodes u or w.
- *parallel reduction*: Let $e_a=(u, v)$ and $e_b=(u, v)$ be two parallel edges in D. D' is obtained by replacing e_a and e_b with a single edge $e_c=(u, v)$ such that $p_{e_c}=1-q_{e_a}*q_{e_b}$ (or $p_{e_c}=p_{e_a}+p_{e_b}-p_{e_a}*p_{e_b}$).

Based on these reliability-preserving reduction, an algorithm to compute the DPR, R_T , of tree of concentrator with no replicated data files and no replicated programs is described as follows:

```
Algorithm TOC (Tree_of_Concentrator )
1. Do reductions as possible as one can on the current
   graph to reduce the space of the graph.
2. Apply Algorithm Connected_Components to every
   interested tree j of concentrator to get V.Component of
   tree j.
3. For every interested tree j ,
   Link_of_tree_j := V.Component of tree j- 1
4. Tree reliability R_T is $\prod_j (P_e)^{Link\_of\_tree_j}$
```

```
Algorithm Connected_Components (D)
   Input :  D = ( V , E )
   output:  V.Component is set to the number of the
            component containing V , for every vertex V.
   begin
      Component_Number : = 1 ;
      While there is an unmarked vertex V do
      Depth_First_Search ( D , V ) ;
      ( Using the following prework :
      V.Component : = Component_Number ; )
      Component_Number : = Component_Number + 1
   end ;
```

We use an example to illustrate how to derive tree reliability.

Example 2 : A distributed computing system with three unique programs and six unique data files distributed in a ten processors C-R network as shown in Fig. 5. Let $p_x = 1$, $p_e = 0.9$, and let P1 need data files F_1, F_3, and F_5 to complete execution.

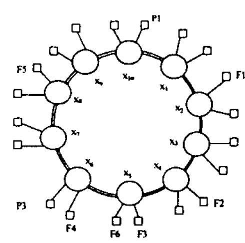


Fig. 5 an example for tree of concentrator

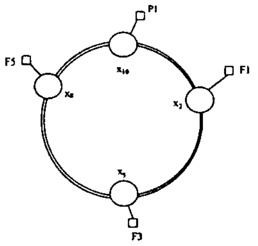


Fig. 6 snapshot of Fig. 5

The analysis steps of the algorithm TOC is:

1. Do reduction as possible as one can on the current graph --- we get snapshot of Fig.6.
2. Apply **Algorithm Connected_Components** to get V.Component of tree j.
3. We get X5 tree: V.Component = 2 and link_of_tree= 2-1=1
   X2 , X8 , and X10 tree : V.Component = 2 and link_of_tree= 2-1=1
4. Tree Reliability R_T=(0.9)(0.9)(0.9)(0.9)=0.6561

3.4 Ring of Trees

A ring of tree topology combines a dual ring with attached trees connected to root station concentrator in the dual ring. In a ring of tree with N terminal stations connected to M stations in the C-R dual ring there are at most M trees attached to the dual ring at root concentrator. Suppose the K stations of interest are located in X distinct trees, $X \leq M$. If all K stations of interest are located in the same tree and share a common root concentrator in the dual ring, then computing the DPR on ring of tree, R_{RT}, is the same as computing R_T for a single tree. otherwise, compute R_{RT} as following step:

Step 1: for each tree j , $1 \leq j \leq X$, compute the reliability of each tree j and get $R_T = \prod_j$ (the reliability of each tree j)

Step 2: compute X-reliability R^x_{C-R} for the X root concentrators of interest in the dual ring.

step 3: combine the results of steps 1 and 2 to obtain

$$R_{RT}=(R^x_{C-R}) (R_T)$$

We use example 3 to illustrate it.

Example 3: A distributed computing system with 3 unique programs and 6 unique data files distributed in a 10 processors C-R network as shown in Fig. 5. Let $p_x = 1$, $p_e = 0.9$, and let P1 need data files F_1, F_3, and F_5 to complete execution.

From the results of example 1 and example 2, we can get R^x_{C-R} and R_T. Then

$R_{RT}=(R^x_{C-R}) (R_T) = (0.7714929) (0.6561)=0.5061764$

4 THE DISTRIBUTED PROGRAM RELIABILITY ANALYSIS WITH REPLICATED DATA FILES AND REPLICATED PROGRAMS

In a DCS, there are frequently duplicated data files and programs, such as in data base systems. Hence we will focus on ring networks with replicated data files and programs in this section. When we analyze the DPR of dual ring, we can first factor the dual ring to be a path network. Then apply the result of path network to the dual ring network. In the ring of tree topology, we can't simply cut the topology into two parts as shown in section 3. We will show that computing DPR on a ring of tree topology is NP-hard.

4.1 A Polynomial Time Algorithm for Computing DPR Over a DCS with a Linear Structure

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 $(v_0, e_1, v_1, e_2, ..., v_{n-1}, e_n, v_n)$ is given. For $1 \leq i \leq n$, 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 the DPR of a DCS with a linear structure D with $|E|=n$ edges, $R(D_H)$, can be state as $Pr(E_n)$. The following theorem provides a recursive method for computing the DPR of a linear DCS, $Pr(E_n)$.

Theorem 2:

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

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

Proof. See reference [8].

Before applying the Theorem 2, we use the following procedure, COMGQ, to compute the values of g_i and Q_i, for $1 \leq i \leq n$, for a given linear DCS with $|E|=n$ edges.

Procedure COMGQ

```
// this procedure computes the values of $g_i$ and $Q_i$, for $1 \leq i \leq n$.//
//h (head) and t (tail) are two indexes moving among nodes. $NF_i$
  is the total number of files 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 $2 \leq i \leq n$ 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 $t \leq n$ 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 $1 \leq i \leq n$ do output($g_i$, $Q_i$) repeat
end COMGQ
```

Now, using the procedure COMGQ and Theorem 2, we are able to provide an algorithm for computing the reliability of a DCS with a linear structure.

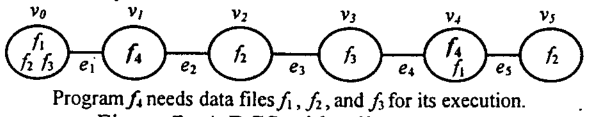
Algorithm Reliability_Linear_DCS(D)
// Given a DCS with a linear structure $D=(V, E, F)$ with $|E|=n$ and a specified set of files H ,//
// this algorithm returns the DPR of D //
Step 1: Call COMGQ to compute the values of g_i and

Q_i, $1 \leq i \leq n$.

Step 2: Evaluate $Pr(E_n)$, recursively using Theorem 2.
Step 3: Return $(Pr(E_n))$.
end Reliability_Linear_DCS

Complexity of the algorithm

For step 1, the computational complexity of the procedure COMGQ is $O(|E||F|)$, where $|E|=n$ and $|F| \geq max((max_{i=0}^{n}(FA_i), H))$, 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 2, $Pr(E_i)$ 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 the 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.



Program f_4 needs data files f_1, f_2, and f_3 for its execution.
Figure 7. A DCS with a linear structure

Example 4:
Consider the linear DCS $D=(V, E, F)$ shown in Figure 7. We have $V=\{v_0, v_1, v_2, v_3, v_4, v_5\}$, $E=\{e_1, e_2, e_3, e_4, e_5\}$ and $F=\{f_1, f_2, f_3, f_4\}$. Let $H=\{f_1, f_2, f_3, f_4\}$. 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:
$Pr(E_1) = Pr(E_2) = Pr(E_3) = q_0Q_1 = p_1$
$Pr(E_4) = Pr(E_3)+(1-Pr(E_0))q_1Q_2+(1-Pr(E_1))q_2Q_3$
$= p_1+q_1p_2p_3p_4+q_1q_2p_3p_4$
$Pr(E_5) = Pr(E_4)+(1-Pr(E_2))q_3Q_4$

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

4.2 A Polynomial Time Algorithm for Computing DPR Over a Dual Ring Network

A circular 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 circular structure. By factoring theorem, the DPR of D can be given as

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

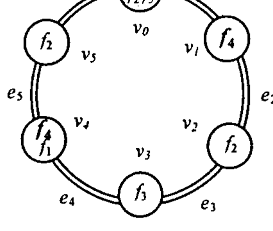
where

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

```
Algorithm Reliability_Circular_DCS(D)
// Given a DCS with a circular 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 $V$ that 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_Circular_DCS($D+e$).
Step 5: Return ($p_e*R_r+q_e*R_l$).
end Reliability_Circular_DCS
```

Example 5: Consider the DCS with a circular topology shown in Figure 8. This is a simply the DCS shown in Figure 7 with one edge e_6 added between nodes V_5 and V_0.



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

Figure 8. A DCS with a ring structure

Applying algorithm Reliability_Circular_DCS, we have

$$R(D_H)=q_6R((D-e_6)_H)+p_6R((D+e_6)_H)$$
$$=q_6R((D-e_6)_H)+p_6\{q_5R((D+e_6-e_5)_H)+p_5R((D+e_6+e_5)_H)\}$$

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 4, 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

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

4.3 The DPR Problem on Ring of Tree Topology with Replicated Data Files and Replicated Programs

In this case, we consider that two or more the same data files and programs can exist in the ring of tree topology. We can't simply cut the topology into two parts as shown in section 3 for DPR analysis. The computation complexity for this case is very difficult to obtain. In fact, we will show the DPR analysis of this case, replicated data files and programs, of ring of tree topology is NP-hard. We use the strategy, transform known NP-hard problem to our reliability problem, to prove DPR problem is NP-hard [10-14].

We first state some *NP*-hard problems as follows.

Theorem 3. Computing DPR for a DCS with a star topology even with each $|FA_i|=2$ is *NP*-hard.

Proof. See reference [8,9].

Theorem 4. Computing DPR for a DCS with a star topology even when there are only two copies of each file is *NP*-hard .

Proof. See reference [8,9].

Theorem 5. Computing DPR for a DCS with a tree topology is *NP*-hard.

Proof. See reference [8,9].

Now, We use the results of Theorem 3, Theorem 4, and Theorem 5 to prove the DPR problem on ring of tree topology is NP-hard.

Theorem 6: 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 $E=\{(s, v_i) \mid 1\le i\le n\}$ with a star topology. We construct a DCS graph $D'=(V', E', H)$ from graph D. where

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

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

It is easy to see that D' is a ring of tree topology with one level of tree. If we assume all added edges, $\{(s_j, s_{j+1})$ and $(s_n, s_1) \mid 1\le j\le n\}$, of D' be perfect reliability , then we have $R(D_H)=R(D'_H)$ for any given $H\subseteq H$. By Theorem 3 and 4, 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 7: Computing DPR for a DCS with a ring of tree topology, in general, is NP-hard.

Proof. By Theorem 6, 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 6, we construct a ring of tree topology with a tree topology. By Theorem 5, 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. □

4.4 An algorithm for computing DPR on a ring of tree topology

In section 4.3 we have shown that computing DPR for a DCS with a ring of tree topology is NP-hard. Now we propose an algorithm, FREA (Fast Reliability Evaluation Algorithm), to compute the DPR on a ring of tree topology. FREA is an algorithm for the reliability evaluation of a DCS with perfect nodes. The FREA algorithm is based on the generalized factoring theorem with employing several reliability-preserving reductions to reduce the size of computed graphs and to simplify the reliability computation.

4.4.1 The Generalized Factoring Theorem

The factoring theorem of network reliability is the basis for a class of algorithms for computing *K*-terminal reliability [15]. This theorem consists of picking an edge of the graph and decomposing the original problem with respect to the two possible states of the edge:

$R(D_H)=p_e R(D_H \mid e \text{ working})+q_e R(D_H \mid e \text{ failed})$ (Eq. 2)

which, for an undirected graph with perfect nodes, can be written

$R(D_H)=p_e R(D_H+e)+q_e R(D_H-e)$ (Eq. 3)

Thus, equation (Eq.3) can be generalized in the following manner.

$R(D_H)=p_{e_1} R(D_H+e_1)+q_{e_1} p_{e_2} R(D_H-e_1+e_2)+\dots$
$+ q_{e_1} q_{e_2} \dots q_{e_{d-1}} p_{e_d} R(D_H-e_1-e_2-\dots-e_{d-1}+e_d)$ (Eq. 4)

, where $\{e_1, e_2, \dots, e_d\}$ is the set of edges incident to the nodes containing the programs being executed.

Equation (Eq. 4) can be recursively applied to the induced graph until either 1) the further induced graph with a node contains all data files needed for the programs to be executed, or 2) the further induced graph contains no FSTs. The induced graph of the former case represents success (reliability=1); the latter case represents failure (reliability=0). Since the subgraphs generated using equation (Eq. 4) will be completely disjoint, no duplicated subgraphs will be generated during the expansion of the computation tree.

4.4.2 Basic Reduction Methods

In order to reduce the size of graph D and therefore reduce the state space of the associated reliability problem, reliability-preserving reductions can be applied. Some reductions are designed and developed to speed up the reliability evaluation. We use four kinds of reduction method in this algorithm that are degree-1 reduction, parallel reduction, series reduction, and degree-2 reduction. These reduction methods are described in section 3.3.

4.4.3 FREA Algorithm

FREA uses equation (Eq. 4) and the basic reliability-preserving reductions discussed in section 3.3 to compute the reliability of DCS. The complete FREA algorithm is stated below.

```
Algorithm FREA(D, H)
Input
D=(V, E, F): the DCS graph D with node set E, edge set V and
file distribution F
H: the set of needed files to be connected
Output:
R(D_H): the distributed program reliability
begin
Step 1: The checking step
if there exists one node $x_i$ such that $FA_i \supseteq H$ then return (1)
endif
if there are no FSTs in D then return (0) endif
Step 2: The reduction step for D
    repeat
      Perform degree-1, parallel, series, and degree-2 reductions
    Until no reductions can be made
Step 3: The formulating step for equation (Eq. 4)
    $D' \leftarrow$ the new graph after the above reduction
    $D3 \leftarrow D2 \leftarrow D'$   /* D3 and D2 are temporary variables for
                      graph D' */
    $R \leftarrow 0$      /* set reliability to 0 */
    $C \leftarrow 1$      /* the constant terms in equation (Eq. 4) */
    for all $e_i \in$ the set of edges incident on the nodes
         containing the executing programs do
         $C \leftarrow C*p_{e_i}$
         $R \leftarrow R+C*\text{FREA}(D3+e_i, H)$
         $C \leftarrow C*q_{e_i}$
         $D2 \leftarrow D3-e_i$
         $D3 \leftarrow$ the new graph after deleting irrelevant
                   components from D2
         if there are no FSTs in D3 then return(R) endif
         repeat
         return(R)
end FREA
```

Example 6: Consider the distributed computing system in Fig. 9. Using FREA to evaluate the reliability of program P1, which need data files F_1, F_2, and F_3 for its execution, we generate the subgraphs shown in Figure 10.

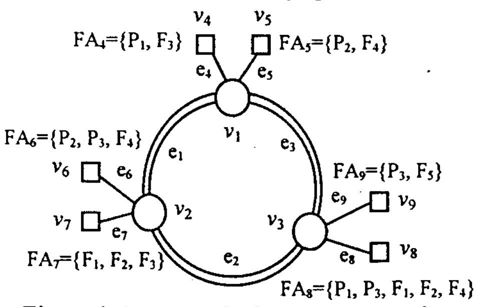


Figure 9 An example for FREA to evaluate DPR

The DPR of program P1 can be computed as follows:

$$DPR_1=p_{e_4}*(p_{e_1}*p_{e_7}+p_{e_1}*q_{e7}*p_{e11}+q_{e1}*p_{e3}*p_{e12}+ q_{e1}*p_{e3}*q_{e12}*p_{e8}+q_{e1}*q_{e3}*p_{e13})+q_{e4}*p_{e10}$$

$$p_{e10}=(1-((1-p_{e_1}*p_{e_2})*q_{e3}))*p_{e7}*p_{e}$$

$$p_{e11}=(1-q_{e2}*q_{e3})*p_{e8}$$

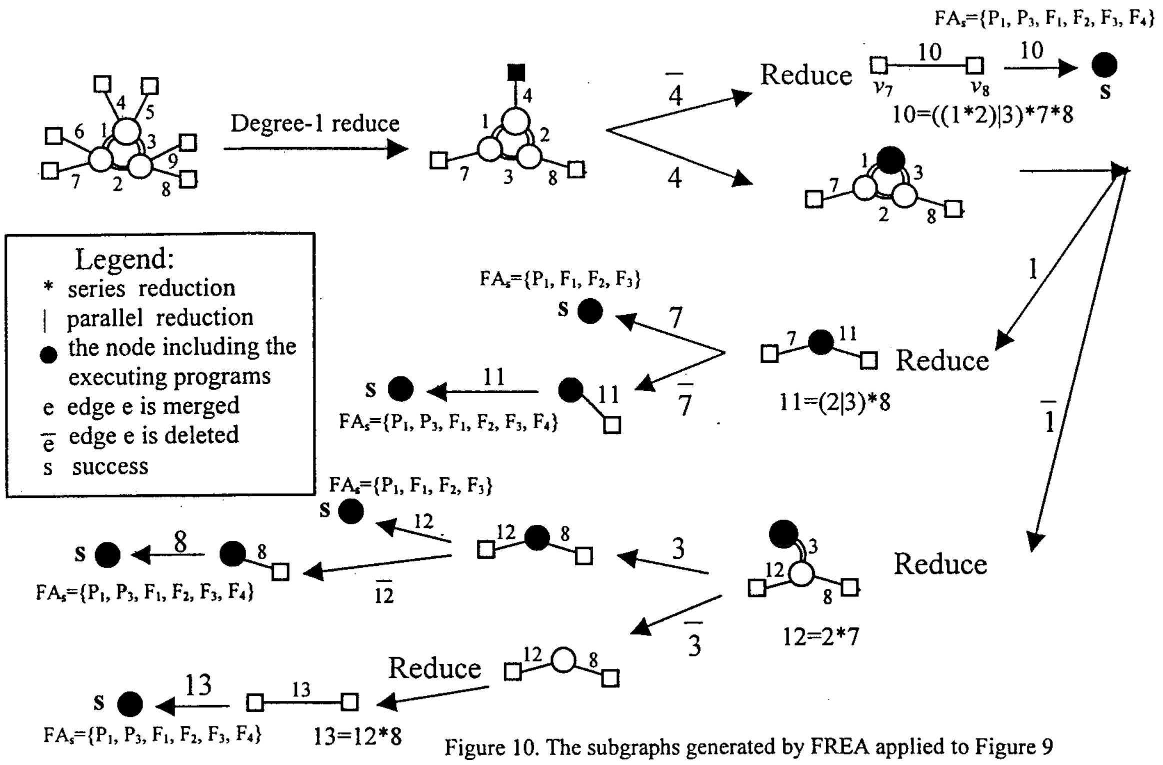

Degree-1 reduce
Reduce
10=((1*2)|3)*7*8
FA$_s$={P$_1$, P$_3$, F$_1$, F$_2$, F$_3$, F$_4$}
11=(2|3)*8
FA$_s$={P$_1$, F$_1$, F$_2$, F$_3$}
12=2*7
13=12*8
Legend:
* series reduction
| parallel reduction
● the node including the executing programs
e edge e is merged
$\bar{e}$ edge e is deleted
s success

Figure 10. The subgraphs generated by FREA applied to Figure 9

$p_{e12} = p_{e2}*p_{e7}$ $\quad$ $p_{e13} = p_{e12}*p_{e8}$

We substitute p_{e10}, p_{e11}, p_{e12}, and p_{e13} to DPR$_1$. Then DPR$_1$ is

$$DPR_1=p_{e4}*(p_{e1}*p_{e7}+p_{e1}*q_{e7}*(1-q_{e2}*q_{e3})*p_{e8} +q_{e1}*p_{e3}*p_{e2}*p_{e7}+q_{e1}*p_{e3}*(1-p_{e2}*p_{e7})*p_{e8}+q_{e1}*q_{e3}* p_{e2}*p_{e7}*p_{e8}) + q_{e4}*(1-((1-p_{e1}*p_{e2})*q_{e3}))*p_{e7}*p_{e8}$$

Let the probability of any link being operational be 0.9. Then the DPR$_1$ is computed as 0.966654.

5. CONCLUSIONS

Distributed Computing Systems (DCSs) offer benefits such as increased reliability, resource sharing, and so on. The time complexity of evaluating the DPR for general topologies is non-polynomial. Fortunately, it is polynomial for ring networks, which are very popular, especially counter-rotating ring networks.

In this paper, We propose polynomial-time algorithms to analyze the DPR of dual ring topology. We also show that solving the DPR on a ring of trees topology is *NP*-hard. Finally we propose FREA algorithm for computing the DPR on a ring of tree topology.

REFERENCE

[1] 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.

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

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

[4] V. K. Prasnna Kumar, S. Hariri and C.S. Raghavendra, " Distributed Program Reliability Analysis", *IEEE Trans. Software Eng.*, Vol. SE-12, No-1, pp.42-50, Jan. 1986.

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

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

[7] 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.

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

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

[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] A. Satyanarayana, M. K. Chang, "Network reliability and the factoring theorem", *Networks*, vol 13, 1983, pp 107-120.