Distributed-Program Reliability Analysis: Complexity and Efficient Algorithms

M. S. Lin
Tamsui Oxford University College, Taipei
M. S. Chang
Chunghwa Telecommunication Training Institute, Taipei
D. J. Chen, Member IEEE
National Chiao-Tung University, Hsin-Chu

Key Words — Distributed computing system, Distributed-program reliability, Computation complexity.

Summary & Conclusions — This paper investigates the problem of distributed-program reliability in various classes of distributed computing systems. This problem is computationally intractable for arbitrary distributed computing systems, even when it is restricted to the class of star distributed computing systems. One solvable case for star distributed computing systems is identified, in which data files are distributed with respective to a consecutive property; a polynomial-time algorithm is developed for this case. A linear-time algorithm is developed to test whether or not an arbitrary star distributed computing system has this consecutive file distribution property. Efficient algorithms may still be sought for computing lower & upper bounds on the distributed program reliability for arbitrary distributed computing systems.

1. INTRODUCTION

Acronyms & Abbreviations[1]

DCS	distributed computing system
DPR	distributed-program reliability
KTR	K-terminal reliability
#EC	number of edge covers
FST	file spanning tree

Definitions (Star DCS)

· Consecutive file distribution. A star DCS D has the consecutive file distribution property iff its nodes can be linearly ordered such that, for each distinct file f_d, the nodes containing file f_d occur consecutively. More formally, a star DCS D has the consecutive file distribution property iff there exists a permutation Π (see *Notation*)

· File cutset. A set C_d of edges of D is a file cutset for file f_d if it consists of all edges (s, v_i) such that node v_i contains file f_d, *ie*, $C_d = \{(s, v_i) : f_d \in A_i\}$.

· Minimal file cutset. A file cutset C is minimal if there is no other file cutset C' such that $C' \subseteq C$. Without loss of generality, reorder the minimal file cutsets, if necessary, by their minimal component: *ie*, for two distinct minimal file cutsets C_i, C_j, $i < j$ iff $\min\{k : (s, v_{\pi(k)}) \in C_i\} < \min\{k : (s, v_{\pi(k)}) \in C_j\}$. ◄

A DCS, in general, is one in which the computing functions are distributed among several physically distinct computing elements [4]. These elements or resources (*eg*, processing elements, data files, programs) can be geographically separated or co-located. Thus, each program can run on one or more computers and can frequently access files stored in other sites. Banking systems, travel agency systems, and power control systems are a few examples of a distributed computing environment [15]. There are many measures to evaluate the performance of DCS. Reliability is an important issue [5]. For traditional networks, many reliability indices have been proposed, *eg*, 2-terminal reliability, all-terminal reliability, and $K-$terminal reliability [1, 7, 12, 13, 18]. However, these measures do not apply to practical DCS since the reliability measure for DCS should capture the effects of redundant distribution of data files.

Kumar *et al* [8, 9] introduced the new reliability measure: DPR to model accurately the reliability of DCS. DPR is defined as the probability that a program with distributed files can run successfully in spite of some faults occurring in the communication edges. A model used to represent such situations is a probabilistic graph. A probabilistic graph has a collection of nodes representing the processing elements (sites) which contain some data files and programs, together with a collection of edges representing communication links. Each edge fails s-independently with known failure probability. For example, consider a possible DCS of a banking system [8, 15] shown in figure 1. Each local disk stores some of the following information:

· consumer accounts file (CAF),
· automated teller machine accounts file (TAF),
· administrative aids file (ADF),
· interest and exchange-rates file (IXF).

[1]The singular & plural of an acronym or abbreviation are always spelled the same.

0018-9529/99/$10.00 ©1999 IEEE

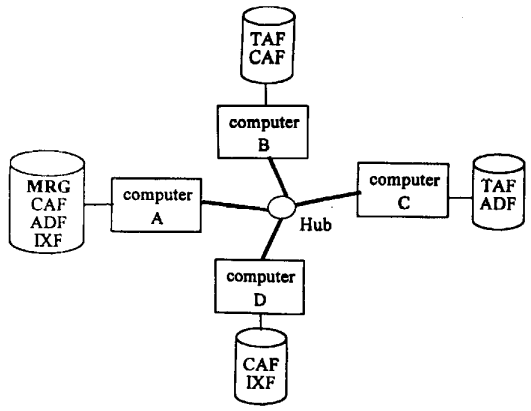

Figure 1: A Distributed Banking System

Management report generation (MRG) in computer A 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 (program) MRG requires data files CAF, TAF, ADF, IXF to complete its execution,
- MRG is running at node v_1, which holds data files CAF, ADF, IXF.

Hence, MRG must access TAF, which is stored in both nodes v_2, v_3. Therefore, the DPR of MRG in figure 2 can be formulated as:

DPR = Pr{(v_1, v_2 are connected) OR (v_1, v_3 are connected)}.

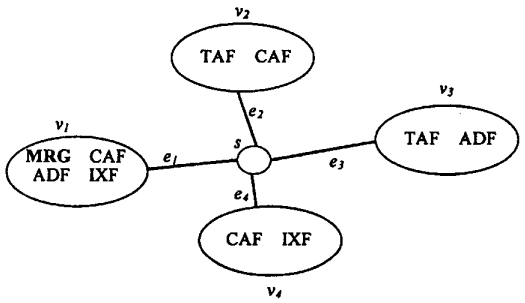

Figure 2: Graph-Model for the Distributed Banking System in Figure 1

Most network reliability problems (*eg*, $K-$terminal reliability) are $\#P-$complete. The class of $\#P-$complete problems was introduced by Valiant [16]. The class $\#P$ contains those problems that involve counting the accepting computations for problems in NP; the class of $\#P-$complete problems contains the hardest problems in $\#P$. As widely recognized, all known exact algorithms for these problems have exponential time complexity, thereby making it unlikely that efficient (polynomial time) algorithms can be developed for this class of problems. Clearly, computing the DPR for general DCS is also $\#P-$complete. This complexity can be averted by considering only a restricted class of DCS. Our objective is to examine the boundary of problem classes separating the polynomially solvable cases from the $\#P-$complete cases. However, polynomial-time algorithms have been developed for computing the DPR over the DCS with linear & ring topologies [10].

Classes of interest here include:

- star,
- 2-tree,
- series-parallel,
- planar topologies.

Section 2 shows that most of them continue to be $\#P-$complete. Section 3 presents a polynomially solvable case of the DPR problem for star topologies in which data files are restricted to a certain type of distribution. Section 4 shows a linear-time algorithm to verify whether or not a star DCS has this restricted class of file distribution.

Assumptions

1. The nodes are perfect.
2. The edges are s-independent and either function or fail with known probabilities. ◀

Notation (General[2])

G	a general graph (of a network)
D	a DCS graph
E	set of edges
V	set of nodes
e_i	a component of E
v_i	a component of V
K	subset of V
A_i	set of files available at v_i
p_i	Pr{e_i functions}
q_i	Pr{e_i fails}: $1-p_i$
f_i	data file i

Notation (Star DCS)

D	a star DCS with $n+1$ nodes, $\{s, v_1, v_2, \ldots, v_n\}$ and n edges $\{(s,v_1),(s,v_2),\ldots,(s,v_n)\}$
n	number of edges in D
e_i	edge (s, v_i), $1 \le i \le n$
m	number of distinct files in D
t	total number of files in D
A_j^{-1}	set of indexes of nodes which contain f_j
Π	$[\pi(1), \pi(2), \ldots, \pi(n)]$: a permutation of numbers $\{1, 2, \ldots, n\}$ such that if $f_d \in A_{\pi(i)}$ and $f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ for all k, $i < k < j$
Φ	ordered set of all minimal file cutsets according to their minimal components
r	number of minimal file cutsets in Φ
C_i	$H(\alpha_i, \beta_i)$: minimal file cutset $\#i$ in Φ; $1 \le i \le r$

[2]Other, standard notation is given in "Information for Readers & Authors" at the rear of each issue.

α_i $\min[k: e_{\pi(k)} \in C_i]$: index of the minimal component in C_i; $1 \le i \le r$

β_i $\max[k: e_{\pi(k)} \in C_i]$: index of the maximal component in C_i; $1 \le i \le r$

$H(i,j)$ $\{e_{\pi(i)}, e_{\pi(i+1)}, \ldots, e_{\pi(j)}\}$; $1 \le i \le j \le n$

$X(i,j)$ event: all edges in $H(i,j)$ fail

W_i $\bigcup_{j=1}^{i} X(\alpha_j, \beta_j)$; the DPR of D can be expressed as 1 - Pr$\{W_r\}$

F_i event: the star DCS D' fails; it consists of $i+1$ nodes $s, v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(i)}$, and i edges $e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(i)}$

$\overline{W}$ complement of event W

2. COMPUTATIONAL COMPLEXITY OF DPR PROBLEM

This section assumes that the reader is familiar with the basic notions of NP-completeness; [6] has an excellent exposition of the theory of NP-completeness. The strategy to show that a DPR problem L_1 is $\#P-$complete is:

1. Pick a problem L_2 already known to be $\#P-$complete.

2. Show how to obtain (in polynomial time) an instance I_1 of L_1 from any instance I_2 of L_2 such that from the solution of I_1 we can determine (in polynomial time) the solution to instance I_2 of L_2.

3. Hence, if we have a polynomial-time algorithm for the DPR problem L_1, then we can obtain polynomial-time algorithms for L_2 using step 2.

4. Conclude that L_1 is $\#P-$complete. ◄

Two known $\#P-$complete problems are:

· KTR [12]
Input: An undirected graph $G = (V, E)$. A set $K \subseteq V$ is distinguished with $|K| \ge 2$.
Output: $R(G_K)$, probability that the set K of nodes of G is connected in G.

· #EC [2]
Input: An undirected graph $G = (V, E)$.
Output: The number of edge covers for G
|{EC ⊆ E: each node of G is an end of some edge in EC}|.

· *Theorem* 1. The KTR problem is polynomially reducible to the DPR problem. ◄

The proof is in appendix A.1

By theorem 1, if we have a polynomial-time algorithm for computing the DPR of D, then we can obtain a polynomial-time algorithm for computing the KTR of G using this construction. However, [12] showed that the problem of computing the KTR in general is $\#P-$complete, so computing the DPR in general is also $\#P-$complete. Therefore, corollary #1 is proved.

· *Corollary* 1. Computing the DPR for a general DCS is $\#P-$complete. ◄

· *Corollary* 2. Computing the DPR for a planar DCS is $\#P-$complete. ◄

The proof is in appendix A.2

For the KTR problem, polynomial-time (or linear-time) algorithms have been developed for other restricted networks, such as a star network, a 2-tree network, and a series-parallel network [14]. If there are no replicated files in DCS, *ie*, if there is only one copy of each file in DCS, the DPR problem can be transformed into the equivalent KTR problem in which the K set is the set of nodes that contain the data files needed for the program under consideration. However, data files are usually replicated and distributed in DCS; so these two problems are different. The remainder of this section shows that computing the DPR over a star DCS, a tree DCS, or a series-parallel DCS in general, is still $\#P-$complete.

· *Theorem* 2. The #EC problem is polynomially reducible to the DPR problem over a star DCS. ◄

The proof is in appendix A.3

· *Corollary* 3. Computing the DPR for a star DCS is $\#P$-complete. ◄

The proof is in appendix A.4

Now we show that computing the DPR remains difficult for a 2-tree topology. A 2-tree is defined recursively as follows:

· The complete graph K2 (a single edge) is a 2-tree.

· Given any 2-tree G on $n \ge 2$ nodes, let (v_i, v_j) be an edge of G. Adding a new node v_k and two edges (v_k, v_i) and (v_k, v_j), produces a 2-tree on $n+1$ nodes.

· *Corollary* 4. Computing the DPR for a 2-tree DCS in general is $\#P-$complete. ◄

The proof is in appendix A.5

· *Corollary* 5. Computing the DPR over a series-parallel DCS is $\#P-$complete. ◄

The proof is in appendix A.6

3. POLYNOMIAL-TIME ALGORITHM FOR COMPUTING THE DPR OF STAR DCS

Section 2 shows that computing the DPR over a star DCS is $\#P-$complete. These results imply that polynomial algorithms are unlikely to exist for solving them. However, an efficient algorithm possibly exists for computing the DPR over a star DCS with a certain restricted class of file distribution. This section presents a polynomial-time algorithm for computing the DPR of a star DCS with a consecutive file distribution.

Let D be a star DCS with the consecutive file distribution property. Then, the minimal file cutsets can be ordered by their minimal component, *ie*, for two distinct minimal file cutsets C_i and C_j, $i < j$ iff
$\min[k: (s, v_{\pi(k)}) \in C_i] < \min[k: (s, v_{\pi(k)}) \in C_j]$.
By definition, D fails iff at least one event,
$X(\alpha_i, \beta_i)$, $1 \le i \le r$, occurs.

If $r = 1$, the unreliability of D is,
$\Pr\{W_1\} = \Pr\{X(\alpha_1, \beta_1)\}$.

If $r \geq 2$, the unreliability of D with the first i's file cutsets is:

$$\begin{aligned}\Pr\{W_i\} &= \Pr\{W_{i-1} \cup X(\alpha_i, \beta_i)\} \\ &= \Pr\{W_{i-1}\} + \Pr\{\overline{W}_{i-1} \cap X(\alpha_i, \beta_i)\}, \text{ for } i \leq r \end{aligned} \quad (1)$$

Consider $\overline{W}_{i-1} \cap X(\alpha_i, \beta_i)$, which implies:
· E_1: For each k, $1 \leq k \leq i-1$, at least one edge, $e \in H(\alpha_k, \beta_k) \equiv C_k$ functions;
· E_2: All edges $\in H(\alpha_i, \beta_i) \equiv C_i$ fail.
By event E_2, event E_1 can be rewritten:
· E_1': For each k, $1 \leq k \leq i-1$, at least one edge, $e \in \{H(\alpha_k, \beta_k) - H(\alpha_i, \beta_i)\}$ functions.

A fundamental difficulty in calculating $\Pr\{E_1'\}$ is that events in E_1' are not, in general, disjoint. However, we can define events S_j that are disjoint:
$S_j = \{E_1'$ occurs and edge $e_{\pi(j)}$ is the last good one$\}$, for $\alpha_{i-1} \leq j \leq \alpha_i - 1$. Thus

$$E_1' \cap E_2 = \bigcup_{j=\alpha_{i-1}}^{\alpha_i - 1} (S_j \cap E_2);$$

$$\Pr\{\overline{W}_{i-1} \cap X(\alpha_i, \beta_i)\} = \Pr\{\bigcup_{j=\alpha_{i-1}}^{\alpha_i - 1} (S_j \cap E_2)\} \quad (2)$$

Since the S_j are disjoint events:

$$\Pr\{\bigcup_{j=\alpha_{i-1}}^{\alpha_i - 1} (S_j \cap E_2)\} = \sum_{j=\alpha_{i-1}}^{\alpha_i - 1} \Pr\{S_j \cap E_2\} \quad (3)$$

The event $S_j \cap E_2$, $\alpha_{i-1} \leq j \leq \alpha_i - 1$, can be decomposed into 3 s-independent events:
· no file cutset fails between $e_{\pi(1)}$ and $e_{\pi(j-1)}$,
· $e_{\pi(j)}$ functions,
· all edges between $e_{\pi(j+1)}$ and $e_{\pi(\beta_i)}$ fail.
So,

$$\begin{aligned}&\Pr\{S_j \cap E_2\} = \\ &\quad [1 - \Pr\{F_{j-1}\}] \cdot p_{\pi(j)} \cdot \Pr\{X(j+1, \beta_i)\} \end{aligned} \quad (4)$$

Therefore, according to (1) – (4):

$$\begin{aligned}\Pr\{W_i\} &= \Pr\{W_{i-1}\} \\ &+ \sum_{j=\alpha_{i-1}}^{\alpha_i - 1} [1 - \Pr\{F_{j-1}\}] \cdot p_{\pi(j)} \cdot \Pr\{X(j+1, \beta_i)\}\end{aligned}$$

Theorem 3 is now established.

· *Theorem* 3. For $2 \leq i \leq r$:

$$\begin{aligned}\Pr\{W_i\} &= \Pr\{W_{i-1}\} \\ &+ \sum_{j=\alpha_{i-1}}^{\alpha_i - 1} [1 - \Pr\{F_{j-1}\}] \cdot p_{\pi(j)} \cdot \Pr\{X(j+1, \beta_i)\}\end{aligned} \quad (5)$$

with boundary conditions:

$$\Pr\{W_1\} = \Pr\{X(\alpha_1, \beta_1)\},$$
$$\Pr\{F_k\} = 0, \text{ for } 0 \leq k \leq \beta_1$$

Before applying theorem 3, compute the values of
$\Pr\{X(j+1, \beta_i)\}$ and $\Pr\{F_{j-1}\}$ for
$2 \leq i \leq r$ and $\alpha_{i-1} \leq j \leq \alpha_i - 1$.

By noting that $\alpha_g < \alpha_h$ whenever $g < h$, the recursive formula is:

$$\Pr\{X(j+1, \beta_i)\} = \quad (6)$$

$$\begin{cases} \dfrac{1}{q_{\pi(\alpha_{i-1})}} \cdot \Pr\{X(\alpha_{i-1}, \beta_{i-1})\} \cdot \displaystyle\prod_{k=\beta_{i-1}+1}^{\beta_i} q_{\pi(k)}, & j = \alpha_{i-1} \\ \dfrac{1}{q_{\pi(j)}} \cdot \Pr\{X(j, \beta_i)\}, & \alpha_{i-1} < j \leq \alpha_i - 1 \end{cases}$$

By starting with: $\Pr\{X(\alpha_1, \beta_1)\} = \displaystyle\prod_{k=\alpha_1}^{\beta_1} q_{\pi(k)}$,
successively determine:
$\Pr\{X(\alpha_1 + 1, \beta_2)\}, \Pr\{X(\alpha_1 + 2, \beta_2)\}, \ldots,$
$\Pr\{X(\alpha_2, \beta_2)\}$,
$\Pr\{X(\alpha_2 + 1, \beta_3)\}, \Pr\{X(\alpha_2 + 2, \beta_3)\}, \ldots,$
$\Pr\{X(\alpha_3, \beta_3)\}$,
$\ddots$
$\Pr\{X(\alpha_{r-1} + 1, \beta_r)\}, \Pr\{X(\alpha_{r-1} + 2, \beta_r)\}, \ldots,$
$\Pr\{X(\alpha_r, \beta_r)\}$.

To obtain the $\Pr\{F_{j-1}\}$ in theorem 3, by definition,

$$\Pr\{F_k\} = \begin{cases} \Pr\{W_{i-1}\} & \text{for } \beta_{i-1} \leq k \leq \beta_i - 1 \\ 0 & \text{for } k \leq \beta_1 - 1 \end{cases} \quad (7)$$

Hence, while computing $\Pr\{W_i\}$ by theorem 3, we can also obtain $\Pr\{F_k\}$, for $\beta_{i-1} \leq k \leq \beta_i - 1$.

3.1 A Polynomial-Time Algorithm

The major algorithm-related strategies to compute the DPR of star DCS are outlined. Assume a given star DCS D and the file distributions A_i for each node. By assuming that D has the property of consecutive file distribution, let Π be a permutation of numbers $\{1, 2, \ldots, n\}$ such that if file $f_d \in A_{\pi(i)}$ and $f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ for all k, $i < k < j$. All file cutsets can be enumerated from A_i as follows: If v_i contains f_d, then C_d contains e_i. Then, α_i, β_i values of C_i can be determined from the permutation Π such that:
$\alpha_i = \min[k : e_{\pi(k)} \in C_i]$ and $\beta_i = \max[k : e_{\pi(k)} \in C_i]$.
The next step removes the file cutsets which are not minimal, and rearranges the remaining minimal file cutsets according to their α_i and β_i values. Finally theorem 3, (6), (7) are used to compute the DPR $= 1 - \Pr\{W_r\}$.

Algorithm REL

Input:
a. A star DCS D with
· $n+1$ nodes $\{s, v_1, v_2, \ldots, v_n\}$,
· n edges $\{(s, v_1), (s, v_2), \ldots, (s, v_n)\}$.
/* $e_i \equiv$ edge (s, v_i) */

b. A permutation $\Pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ of numbers $\{1, 2, \ldots, n\}$ such that if $f_d \in A_{\pi(i)}$, $f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ for all k, $i < k < j$.

Output: The DPR of D

Steps:

1. /* find all file cutsets */
   a. /* initialization step */
   For $i \leftarrow 1$ to m Do
   $C_i \leftarrow \emptyset$
   End_For;
   b.
   For $i \leftarrow 1$ to n Do
   For each $f_d \in A_i$ Do
   $C_d \leftarrow C_d \cup \{e_i\}$
   End_For;
   End_For;
2. /* set the values of α_i, β_i for $1 \le i \le m$ */
   For $i \leftarrow 1$ to m Do
   $\alpha_i \leftarrow \min\{k : e_{\pi(k)} \in C_i\}$
   $\beta_i \leftarrow \max\{k : e_{\pi(k)} \in C_i\}$
   End_For;
3. /* find all minimal file cutsets */
   $\Phi \leftarrow \emptyset$;
   For $i \leftarrow 1$ to m Do
   $\Phi \cup \{C_i\}$
   End_For;
   For $1 \le i, j \le m$ Do
   If ($\alpha_i \ge \alpha_j$ and $\beta_i \le \beta_j$)
   Then remove C_j from Φ
   End_If; /* this implies $C_i \subseteq C_j$ */
   End_For;
4. Reorder the minimal file cutsets in Φ for two distinct minimal file cutsets C_i and C_j, $i < j$ iff $\alpha_i < \alpha_j$
5. /* Compute $\Pr\{X(j+1, \beta_i)\}$ for $2 \le i \le r$ and $\alpha_{i-1} \le j \le \alpha_i - 1$, by (6) */
   $$\Pr\{X(\alpha_1, \beta_1)\} \leftarrow \prod_{k=\alpha_1}^{\beta_1} q_{\pi(k)};$$
   For $i \leftarrow 2$ to r Do
   $$\Pr\{X(\alpha_{i-1}+1, \beta_i)\} \leftarrow \frac{1}{q_{\pi(\alpha_{i-1})}} \cdot \Pr\{X(\alpha_{i-1}, \beta_{i-1})\} \cdot \prod_{k=\beta_{i-1}+1}^{\beta_i} q_{\pi(k)};$$
   For $j \leftarrow \alpha_{i-1} + 2$ to $\alpha_i - 1$ Do
   $$\Pr\{X(j+1, \beta_i)\} \leftarrow \frac{1}{q_{\pi(j)}} \cdot \Pr\{X(j, \beta_i)\}$$
   End_For;
   End_For;
6. /* Apply theorem 3 and (7) to compute $\Pr\{W_i\}$ and $\Pr\{F_j\}$ */
   a. /* boundary conditions */
   $\Pr\{W_1\} \leftarrow \Pr\{X(\alpha_1, \beta_1)\}$;
   For $k \leftarrow 0$ to $\beta_1 - 1$ Do
   $\Pr\{F_k\} \leftarrow 0$
   End_For;
   b.
   For $i \leftarrow 2$ to r Do
   For $k \leftarrow \beta_{i-1}$ to $\beta_i - 1$ Do
   $\Pr\{F_k\} \leftarrow \Pr\{W_{i-1}\}$
   End_For;
   $$\Pr\{W_i\} \leftarrow \Pr\{W_{i-1}\} + \sum_{j=\alpha_{i-1}}^{\alpha_i - 1} [1 - \Pr\{F_{j-1}\}] \cdot p_{\pi(j)} \cdot \Pr\{X(j+1, \beta_i)\}$$
   End_For;
7. DPR $\leftarrow 1 - \Pr\{W_r\}$;
   Output(DPR);

End Algorithm REL

3.2 Complexity Analysis

The time complexity of Algorithm REL is analyzed.

Step 1 takes

$$O\left(m + \sum_{i=1}^{m} |A_{\pi(i)}|\right) = O(m + t) = O(t)$$

time (since $m < t$) to identify all file cutsets; t denotes the total number of files in D.

Step 2 takes

$$O\left(2 \cdot \sum_{i=1}^{m} |C_i|\right) = O(t)$$

time to set α_i and β_i, $1 \le i \le m$.

Step 3 takes $O(m^2)$ time to obtain all minimal file cutsets.

Step 4 requires the reordering of all minimal file cutsets in a nondecreasing order of their index of the minimal component. This ordering can be executed in $O(r \cdot \log(r))$ using an efficient sorting algorithm; r denotes the number of minimal file cutsets.

Step 5 evaluates $\Pr\{X(j+1, \beta_i)\}$ by using (6); this requires that

$$O\left(\sum_{i=2}^{r} [\beta_i - \beta_{i-1} + 2]\right) = O(\beta_r - \beta_1 + r) \approx O(n + r),$$
for $j = \alpha_{i-1}$

$$O\left(\sum_{i=2}^{r} (1)\right) = O(r - 1) = O(r), \text{ for } \alpha_{i-1} \le j \le \alpha_i - 1$$

Hence, the total time to evaluate all $\Pr\{X(j+1, \beta_i)\}$ is $O(n + r)$.

Step 6 takes

$$O\left(\sum_{i=2}^{r} [\beta_i - \beta_{i-1}]\right) = O(\beta_r - \beta_1) \approx O(n)$$

to compute all $\Pr\{F_k\}$; and takes

$$O\left(\sum_{i=2}^{r} [1 + 3 \cdot (\alpha_i - \alpha_{i-1})]\right) = O(1 + 3 \cdot [\alpha_r - \alpha_1]) \approx O(n)$$

to compute all $\Pr\{W_i\}$. Therefore, the total time is $O(n)$.

Step 7 performs in constant time.
Thus the entire algorithm has time complexity

$$O(t + t + m^2 + r \cdot \log(r) + (n + r) + n).$$

Since $t \leq m \cdot n$, and $r \leq m$, the time complexity of Algorithm REL is $O(m^2 + m \cdot n)$.

3.3 An Application of Algorithm REL

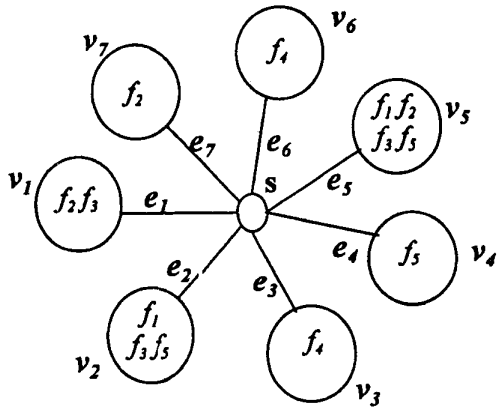

Figure 3: Star DCS with Consecutive-File-Distribution

Consider the star DCS in figure 3; it has the consecutive file distribution property and the associated permutation $\Pi = [3, 6, 4, 2, 5, 1, 7]$; section 4 shows how to identify the associated permutation when the star DCS has the consecutive file distribution property.

Procedure

1. Find the file cutsets:
   $C_1 = \{e_2, e_5\}$, $C_2 = \{e_1, e_5, e_7\}$, $C_3 = \{e_1, e_2, e_5\}$,
   $C_4 = \{e_3, e_6\}$, $C_5 = \{e_2, e_4, e_5\}$.
2. According to the permutation:
   $\pi(1) = 3$, $\pi(2) = 6$, $\pi(3) = 4$, $\pi(4) = 2$, $\pi(5) = 5$,
   $\pi(6) = 1$, $\pi(7) = 7$;
   and to the results of step 1:
   $\alpha_1 = 4$, $\beta_1 = 5$, $\alpha_2 = 5$, $\beta_2 = 7$, $\alpha_3 = 4$, $\beta_3 = 6$,
   $\alpha_4 = 1$, $\beta_4 = 2$, $\alpha_5 = 3$, $\beta_5 = 5$.
3. Because $C_1 \subset C_3$ and $C_1 \subset C_5$, remove C_3 and C_5.
   Thus, the set of minimal file cutsets is: $\Phi = \{C_1, C_2, C_4\}$.
4. Reorder the minimal file cutsets in so that for,
   C_i, C_j, $i < j$ iff $\alpha_i < \alpha_j$:
   $C_1 = \{e_3, e_6\}$, $\alpha_1 = 1$, $\beta_1 = 2$,
   $C_2 = \{e_2, e_5\}$, $\alpha_2 = 4$, $\beta_2 = 5$,
   $C_3 = \{e_1, e_5, e_7\}$, $\alpha_3 = 5$, $\beta_3 = 7$.
5. Use (6):
   $\Pr\{X(1,2)\} = q_3 \cdot q_6$, $\Pr\{X(2,5)\} = q_6 \cdot q_4 \cdot q_2 \cdot q_5$,
   $\Pr\{X(3,5)\} = q_4 \cdot q_2 \cdot q_5$, $\Pr\{X(4,5)\} = q_2 \cdot q_5$,
   $\Pr\{X(5,7)\} = q_5 \cdot q_1 \cdot q_7$.
6. Use theorem 3 and (7) to compute $\Pr\{W_i\}$ and $\Pr\{F_k\}$ for $2 \leq i \leq 3$ and $\beta_{i-1} \leq k \leq \beta_i - 1$:

Boundary condition
$\Pr\{W_1\} = q_3 \cdot q_6$, $\Pr\{F_0\} = \Pr\{F_1\} = 0$.

$i = 2$:
$\Pr\{F_2\} = \Pr\{F_3\} = \Pr\{F_4\} = \Pr\{W_1\} = q_3 \cdot q_6$,
$\Pr\{W_2\} = \Pr\{W_1\} + [1 - \Pr\{F_0\}] \cdot p_3 \cdot \Pr\{X(2,5)\} (j = 2)$
$+ [1 - \Pr\{F_1\}] \cdot p_6 \cdot \Pr\{X(3,5)\} \quad (j = 3)$
$+ [1 - \Pr\{F_2\}] \cdot p_4 \cdot \Pr\{X(4,5)\} \quad (j = 4)$
$= q_3 \cdot q_6 + p_3 \cdot q_6 \cdot q_4 \cdot q_2 \cdot q_5 + p_6 \cdot q_4 \cdot q_2 \cdot q_5$
$+ (1 - q_3 \cdot q_6) \cdot (p_4 \cdot q_2 \cdot q_5)$

$i = 3$:
$\Pr\{F_5\} = \Pr\{W_2\}$
$\Pr\{W_3\} = \Pr\{W_2\} + [1 - \Pr\{F_3\}] \cdot p_2 \cdot \Pr\{X(5,7)\} (j = 5)$
$= q_3 \cdot q_6 + p_3 \cdot q_6 \cdot q_4 \cdot q_2 \cdot q_5 + p_6 \cdot q_4 \cdot q_2 \cdot q_5 +$
$(1 - q_3 \cdot q_6) \cdot (p_4 \cdot q_2 \cdot q_5) + (1 - q_3 \cdot q_6) \cdot (p_2 \cdot q_5 \cdot q_1 \cdot q_7)$.

7. Therefore, DPR is:
   $\text{DPR} = 1 - \Pr\{W_3\}$
   $= 1 - [q_3 \cdot q_6 + p_3 \cdot q_6 \cdot q_4 \cdot q_2 \cdot q_5 + p_6 \cdot q_4 \cdot q_2 \cdot q_5$
   $+ (1 - q_3 \cdot q_6) \cdot (p_4 \cdot q_2 \cdot q_5) + (1 - q_3 \cdot q_6) \cdot (p_2 \cdot q_5 \cdot q_1 \cdot q_7)]$.

4. LINEAR-TIME ALGORITHM: TESTING FOR THE CONSECUTIVE FILE DISTRIBUTION PROPERTY IN A STAR DCS

Section 3 presented a polynomial-time algorithm for computing the DPR of a star DCS when it has the consecutive file distribution property. This section tests whether or not a star DCS has the consecutive file distribution property. The problem statement is:

Input: A star DCS D with $n + 1$ nodes $s, v_1, v_2, \ldots, v_n$, and file distributions A_i, $1 \leq i \leq n$.
Output: A permutation $\Pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ of numbers $\{1, 2, \ldots, n\}$ such that if $f_d \in A_{\pi(i)}$ and $f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ for all k, $i < k < j$.

A solution does not always exist. To facilitate searching for the correct ordering of Π, use a data structure of a $PQ-$tree [3]. A $PQ-$tree is a rooted tree that has two varieties of nodes: $P-$nodes and $Q-$nodes. A $P-$node is a node whose children can be arbitrarily permuted. A $Q-$node is a node whose children are ordered or reverse-ordered. The frontier of a $PQ-$tree is the permutation of leaves from left to right. Two $PQ-$trees are equivalent iff one can be transformed into the other by applying a sequence of the transformation rules:

- arbitrarily permute the children of a $P-$node;
- reverse the children of a $Q-$node.

The algorithm uses a $PQ-$tree data structure.

Algorithm Check_Consecutive_File_Distribution

Input: A star DCS D with $n + 1$ nodes $s, v_1, v_2, \ldots, v_n$, n edges $e_1, e_2, \ldots, e_n$, where $e_i = (s, v_i)$ for $1 \leq i \leq n$; and file-available set,
$A_i = \{f_j$: for each f_j stored in node $v_i\}$ for $1 \leq i \leq n$.

Output: A permutation $\Pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ of numbers $\{1, 2, \ldots, n\}$ such that if $f_d \in A_{\pi(i)}$ and $f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ for all k, $i < k < j$.

0. $T \leftarrow$ universal tree; /* a single $P-$node
connected to all the leaf nodes of $\{1, 2, \ldots, n\}$ */
1.
For $j \leftarrow 1$ to m Do
$A_j^{-1} \leftarrow \emptyset$
End_For;
For $i \leftarrow 1$ to n Do
For each $f_j \in A_i$ Do
$A_j^{-1} \leftarrow \{i\}$
End_For;
End_For;
For $j \leftarrow 1$ to m Do
$T \leftarrow$ REDUCE(T, A_j^{-1})
End_For;
If T is a null tree
Then print out "D has no consecutive file distribution property"
Else print out the Frontier of T,
End_If;

End Check_Consecutive_File_Distribution

The routine REDUCE attempts to apply a set of 11 templates. Each template consists of a pattern to be matched against the current $PQ-$tree and the set A_j^{-1}, and a replacement to be substituted for the pattern. The templates are applied from the bottom to the top of the tree. The null tree is returned when no template applies. See [3] for details of the algorithm.

Complexity Analysis

For A_j^{-1}, $1 \leq j \leq m$, it can be obtained in $O\left(m + \sum_{i=1}^{n} |A_i|\right)$ steps.
The loop of the REDUCE routine can be computed in $O\left(m + n + \sum_{j=1}^{m} |A_j^{-1}|\right)$ steps [3]. Further,
$\sum_{i=1}^{n} |A_i| = \sum_{j=1}^{m} |A_j^{-1}| = t$ (the total number of files in D).
Therefore, the time complexity for
Check_Consecutive_File_Distribution is:
$O(m+t) + O(m+n+t) = O(m+n+t)$.

Example

Consider the star DCS D in figure 3.
Apply Check_Consecutive_File_Distribution; then:
$A_1^{-1} = \{2,5\}$, $A_2^{-1} = \{1,5,7\}$, $A_3^{-1} = \{1,2,5\}$,
$A_4^{-1} = \{3,6\}$, $A_5^{-1} = \{2,4,5\}$.

Figure 4 displays the reduction steps; for the $PQ-$tree, a $P-$node is drawn as a circle and a $Q-$node as a rectangle. Figure 4 shows that:
· the star DCS D of figure 3 has the consecutive file distribution property;
· one of the associative permutations is:
$\Pi = [3, 6, 4, 2, 5, 1, 7]$.

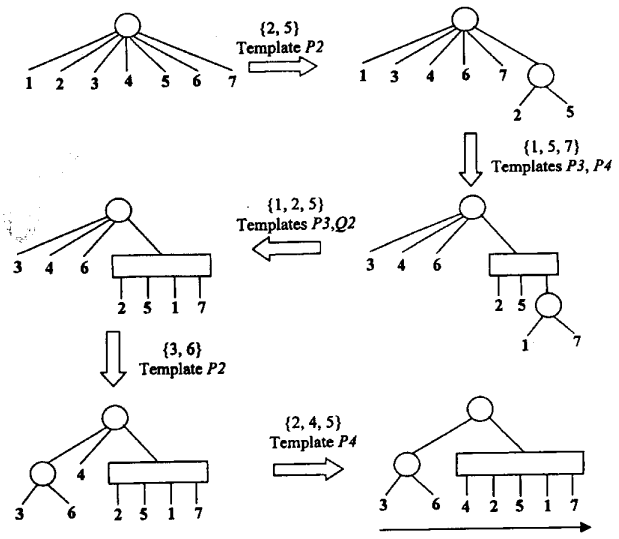

Figure 4: Reduction Steps, Using a $PQ-$Tree

APPENDIX

A.1 Proof of Theorem 1

Let $G = (V, E)$ be a network graph with a subset of nodes $K \subseteq V$. Construct a DCS graph $D = (V, E)$ from G such that v_i of D contains f_i iff $v_i \in K$ in G. All distinct files in D are interconnected iff all nodes of K are connected in G. Also, let each edge of D have the same operational probability as the corresponding edge of G. Then, the DPR of D is equal to the KTR of G. In this case the DCS D can be obtained from G in polynomial time.

A.2 Proof of Corollary 2

The proof of theorem 1 shows that the KTR problem is just a special case of the DPR problem. It has been shown that computing the KTR over a planar network is $\#P-$complete [11]. This also immediately implies that computing the DPR over a planar DCS is still $\#P-$complete.

A.3 Proof of Theorem 2

Given a graph G with n edges $e_1, e_2, \ldots, e_n$, we construct a star DCS D such that the number of edge covers in G can be expressed as a function of the DPR of D. Construct a star DCS $D = (V', E')$ where
$V' = \{s, v_1', v_2', \ldots, v_n'\}$, $E' = \{(s, v_1'), (s, v_2'), \ldots, (s, v_n')\}$
and node v_i' contains files f_g & f_h iff $e_i = (v_g, v_h)$ in G.
We now consider a file spanning tree (FST) T, which is a subgraph of D and its nodes hold all the needed data files:

$$\bigcup_{v_i' \in T} \{f_j : v_i' \text{ contains file } f_j\}$$
$$= \bigcup_{v_i' \in V'} \{f_j : v_i' \text{ contains file } f_j\}.$$

Thus there is a one-to-one correspondence between one of the sets of edge covers in G and one FST in D. The DPR of D can be expressed as:

$$\text{DPR} = \sum_{\substack{\text{for all FST} \\ T \text{ in } D}} \left[\left(\prod_{(v_0, v_i') \in E' - T'} (1 - p_i) \right) \cdot \left(\prod_{(v_0, v_i') \in T'} p_i \right) \right]$$

p_i = reliability of edge (s, v_i') of D, $1 \le i \le n$.

Set $p_i = \frac{1}{2}$, for all $1 \le i \le n$, then:

$$\text{DPR} = \sum_{\substack{\text{for all FST} \\ T \text{ in } D}} \left(\frac{1}{2}\right)^n, \text{ or}$$

$$\text{DPR} \cdot 2^n = \sum_{\substack{\text{for all FST} \\ T \text{ in } D}} 1 = \#\text{of FST in } D$$

$$= \#\text{ of edge covers in } G$$

Since D can be constructed from G in polynomial time, the number of edge covers in G can be solved in polynomial time if we have a polynomial-time algorithm for computing the DPR of D.

A.4 Proof of Corollary 3

This follows from theorem 2 and the fact that #EC have been shown to be $\#P-$complete [2].

A.5 Proof of Corollary 4

Let D be a star DCS with $n+1$ nodes s, $v_1, v_2, \ldots, v_n$, and n edges $(s, v_1), (s, v_2), \ldots, (s, v_n)$. Construct from D a 2-tree DCS D' such that D and D' have the same DPR. Embed the 2-tree DCS D' into the star DCS D by adding some virtual edges (v_i, v_{i+1}), $1 \le i \le n-1$. Now, D' is a 2-tree DCS on $n+1$ nodes. If we stipulate that each virtual edge has operational-probability = 0, then the DPR of D is reduced to the DPR of D'. By corollary 3, since computing the DPR over a star topology in general is $\#P-$complete, then computing the DPR over a 2-tree topology is also $\#P-$complete.

A.6 Proof of Corollary 5

From [17], a 2-tree graph is a maximal series-parallel graph. A maximal series-parallel graph is a series-parallel graph with neither loops nor parallel edges. Since computing the DPR over a 2-tree topology is $\#P-$complete, computing the DPR over a series-parallel DCS is also $\#P-$complete.

REFERENCES

[1] K.K. Agrawal, S. Rai, "Reliability evaluation in computer-communication networks", *IEEE Trans. Reliability*, vol R-30, 1981 Apr, pp 32 - 35.

[2] M.O. Ball, J.S. Provan, D.R. Shier, "Reliability covering problems", *Networks*, vol 21, 1991, pp 345 - 357.

[3] K.S. Booth, G.S. Leuker, "Testing for the consecutive ones property, interval graphs, and graph planarity using $PQ-$tree algorithms", *J. Computing System Science*, vol 13, 1976, pp 335 - 379.

[4] T.C.K. Chou, J.A. Abraham, "Load redistribution under failure in distributed systems", *IEEE Trans. Computer*, vol 32, 1983 Sep, pp 799 - 808.

[5] J. Garcia-Molina, "Reliability issues for fully replicated distributed database", *IEEE Trans. Computer*, vol 16, 1982 Sep, pp 34 - 42.

[6] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979; Freeman.

[7] A.P. Grnarov, M. Gerla, "Multi-terminal reliability analysis of distributed processing system", *Proc. 1981 Int'l Conf. Parallel Processing*, 1981 Aug, pp 79 - 86.

[8] A. Kumar, S. Rai, D.P. Agrawal, "On computer communication network reliability under program execution constraints", *IEEE JSAC*, vol 6, 1988 Oct, pp 1393 - 1399.

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

[10] M.S. Lin, D.J. Chen, "Two polynomial-time algorithms for computing reliability in a linear and a circular distributed system", *PDPTA '97*, 1997 Jun, Las Vegas.

[11] J.S. Provan, "The complexity of reliability computations in planar and acyclic graphs", *SIAM J. Computing*, vol 15, 1986, pp 694 - 702.

[12] A. Rosenthal, "A computer scientist looks at reliability computations in: reliability and fault tree analysis", *SIAM*, 1975, pp 133 - 153.

[13] A. Satyanarayana, J.N. Hagstrom, "A new algorithm for the reliability analysis of multi-terminal networks", *IEEE Trans. Reliability*, vol R-30, 1981 Oct, pp 325 - 334.

[14] A. Satyanarayana, R.K Wood, "A linear-time algorithm for computing $k-$terminal reliability in series-parallel networks", *SIAM J. Computing*, vol 14, 1985 Nov, pp 818 - 832.

[15] D.A. Sheppard, "Standard for banking communication system", *IEEE Trans. Computer*, vol 21, 1987 Nov, pp 92 - 95.

[16] L.G. Valiant, "The complexity of enumeration and reliability problems", *SIAM J. Computing*, vol 8, 1979, pp 410 - 421.

[17] P. Winter, "Steiner problem in networks: A survey", *Networks*, vol 17, 1987, pp 129 - 167.

[18] R.K. Wood, "Factoring algorithms for computing $k-$terminal network reliability", *IEEE Trans. Reliability*, vol R-35, 1986 Aug, pp 269 - 278.

AUTHORS

Dr. Min-Sheng Lin; Dep't of Information Management; Tamsui Oxford University College; 32 Chen Li Rd; Tamsui, Taipei 25103 TAIWAN – R.O.C.
Internet (e-mail): mlin@jupiter.touc.edu.tw

Min-Sheng Lin received his MS (1991) & PhD (1994) in Computer Science & Information Engineering from National Chiao Tung University (HsinChu, Taiwan). He is an Associate Professor at Tamsui Oxford University College. His research interests include reliability and performance evaluation of distributed computing systems.

Ming-Sang Chang; Chunghwa Telecommunication Training Inst; 168 Min Chu Rd; Pan Chiao; Taipei 22077 TAIWAN – R.O.C.
Internet (e-mail): mschang@chtti.com.tw

Ming-Sang Chang received BS (1979) in Electronic Engineering from National Taiwan University of Science and Technology (Taipei, Taiwan) and MS (1982) in Information Engineering from TamKang University (Taipei, Taiwan). He is a PhD candidate in the Dep't of Computer Science & Information Engineering, National Chiao Tung Univ. His research interests include computer network, performance evaluation, distributed system, and reliability evaluation.

Dr. Deng-Jyi Chen; Dep't of Computer Science & Information Eng'g; National Chiao Tung Univ; 1001 Ta Hsueh Rd.; HsinChu 30050 TAIWAN – R.O.C.
Internet (e-mail): djchen@csie.nctu.edu.tw

Deng-Jyi Chen received the BS (1983) in Computer Science from Missouri State University (Cape Girardeau), and MS (1985) and PhD (1988) in Computer Science from the University of Texas (Arlington). He is a Professor at National Chiao Tung University. He has published more than 80 referred journal and conference papers in reliability and performance modeling of distributed systems, computer networks, object-oriented systems, and software reuse. Dr. Chen works very closely with industrial companies and consults for many local (both for software & hardware companies. He has been a chief leader of designing & implementing two commercial products which are now marketing around the world. Dr. Chen has received research awards each year from the National Science Council, Taiwan for the past nine years, and serves as a committee member in several academic & industrial organization.

Manuscript TR98-006 received: 1998 January 21;
revised: 1998 June 20, November 30
Responsible editor: A. von Mayrhauser
Publisher Item Identifier S 0018-9529(99)04899-X