International Conference on
Information, Communications and Signal Processing
ICICS '97
Singapore, 9-12 September 1997

The Reliability Problem in Distributed Database Systems

Min-Sheng Lin
Department of Management Information Systems
Tamsui Oxford University College
32, Chen-Li Rd., Tamsui,
Taipei, Taiwan, R.O.C. 25103
Email:MLIN@JUPITER.TOUC.EDU.TW

and

Deng-Jyi Chen
Institute of Computer Science and Information Engineering
National Chiao-Tung University
Hsin Chu, Taiwan, R.O.C. 30050
Email: DJCHEN@CSIE.NCTU.EDU.TW

Abstract

The reliability of a distributed database systems is the probability that a program which runs on multiple processing elements and needs to communicate with other processing elements for remote database will be executed successfully. This reliability varies according to 1) the topology of the distributed database system, 2) the reliability of the communication links, 3) the databases and program distribution among processing elements, and 4) the databases required to execute a program. This paper shows that solving this reliability problem is *NP*-hard even when the distributed database system is restricted to a series-parallel, a 2-tree, a tree, or a star structure. Two polynomial-time algorithms are proposed for computing the reliability of a distributed program which runs on a linear and a ring distributed database system, respectively.

1 Introduction

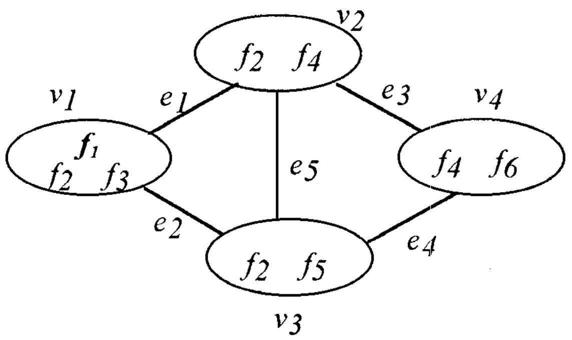
A typical distributed database system (DDBS) consists of processing elements (nodes), communication links (edges), memory units, database, and programs. These resources are interconnected via a communication network that dictates how information flows between nodes. Programs residing on some nodes can run using database at other nodes.

One important issue in the design of a DDBS is reliability. A large amount of work has been devoted to developing algorithms to compute measures of reliability for DDBSs. One typical reliability measure for DDBSs is the K-terminal reliability (KTR). KTR is the probability that a specified set of nodes K, which is subset of all the nodes in the DDBS, remains connected in a DDBS whose edges may fail independently of each other, with a known probabilities. However, the KTR measure is not applicable to practical DDBSs since a reliability measure for DDBSs should capture the effects of redundant distribution of programs and databases.

The distributed program reliability (DPR) was introduced to accurately model the reliability of DDBSs. Consider DDBS in which the nodes are perfectly reliable but the edges can fail, s-independently of each other, with known probabilities. For successful execution of a distributed program, it is essential that the node containing the program, other nodes that have required databases, and the edges between them be operational. DPR is thus defined as the probability that a program with distributed databases can run successfully in spite of some faults occurring in the edges. To illustrate the definition of DPR, consider the DDBS shown in Figure 1. There are four nodes (v_1, v_2, v_3, v_4) and five edges (e_1, e_2, e_3, e_4, e_5). Program f_1 requires databases f_2, f_3, and f_4 to complete execution, and it is running at node v_1, which holds databases f_2 and f_3. Hence, it must access database f_4, which is resident at both node v_2 and node v_4. Therefore, the reliability of distributed program f_1 can be formulated as follows:

DPR(program f_1)=Prob((v_1 and v_2 are connected) or (v_1 and v_4 are connected)).

0-7803-3676-3/97/$10.00 © 1997 IEEE



Program f_1 needs databases f_2, f_3, and f_4 to complete execution.

Figure 1: A simple DDBS

Several algorithms have been proposed for evaluation DPR[1-3]. However, we have seen that none meets our desire for efficient algorithms. At this point, one must conclude that either the approaches examined are just not sufficiently clever, or that no efficient algorithms exist for our reliability problems. Nevertheless, tools in complexity theory do provide a vehicle for giving strong evidence that no polynomial time algorithm exists for certain problems. The purpose of this paper is to show that the DDBS problem, in general, is *NP*-hard even on a DDBS with a series-parallel, a 2-tree, a tree, or a star structure. Two polynomial-time algorithms are proposed for computing the DPR of a DDBS with a linear and a ring structure respectively.

In this paper we will make use of the following notation:

$D=(V, L, F)$	an undirected DDBS graph with vertex set V, link set L, and database set F which is distributed in D
V	the set of vertics that are all perfectly reliable
L	the set of links that can fail, *s*-independently of each other, with known probability
F	the set of databases (including databases and programs) distributed in D
$H \subseteq F$	the specified set of databases that must communicate with each other through the links
$FA_i \subseteq F$	the set of databases available at vertex i
p_i	the reliability of link l_i
q_i	$\equiv 1-p_i$
$R(D_H)$	the DPR of D with H specified
	$\equiv$the probability that all files in H can communicate with each other through the links in D

2 The Computational Complexity of the DPR Problem

A generally accepted method for providing evidence of intractability is to prove *NP*-complete or *NP*-hard. *NP*-hard is not a proof of intractability, but is convincing evidence. An efficient solution for any *NP*-hard problem provides efficient methods for every problem in *NP*, which contains a formidable list of apparently difficult problems.

Computing *K*-terminal reliability was first shown to be *NP*-hard by Rosenthal [4], and it follows from Valiant[5] that the problem is #P-complete even when G is planar. Two special cases of this reliability problem are the most frequently encountered, the terminal-pair problem where $|K|=2$, and all-terminal problem where $K=V$. These problems are also #P-complete [6] in general. However, for the *KTR* problem polynomial-time (or linear-time) algorithms have been developed for other restricted networks, such as linear systems, ring systems, stars, trees, and series-parallel graphs.

Obviously, if there are no replicated databases, i.e., if there is only one copy of each database in the DDBS, then the DPR problem can be transformed into a *KTR* equivalent problem in which the K set is the set of nodes that contain the databases needed for the program under consideration. However, databases are usually replicated and distributed in DDBS, so these two problems are different.

Complexity results are obtained by transforming known *NP*-hard problems into our reliability problems. For this reason, we first state some known *NP*-hard problems below.

1. K-Terminal Reliability (KTR)

Input: an undirected graph $G=(V, L)$, where V is the set of vertics and L is the set of links 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 vertics of G is connected in G.

2. Number of Link Covers (#LC) [7]

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

Output: the number of link covers for G

$\equiv |\{C \subseteq L$: each vertex of G is an end of some link in $C\}|$.

3. Number of Vertex Covers (#VC) [8]

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

Output: the number of vertex covers for G

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

Theorem 1. Computing DPR for a general DDBS is *NP*-hard.

Proof. We reduce the *KTR* problem to our DPR problem. For a given network $G=(V, L)$ and a specified set $K \subseteq V$, we can define an instance of the DPR problem. Construct a DDBS graph $D=(V, L, F)$ in which the topology and the reliability of each link are the same as G. Let $F = \bigcup_{\text{node } V_i \in K} \{f_i\}$ and $FAv_i = \{f_i\}$ if vertex $v_i \in K$, else $FAv_i = \varnothing$ for each vertex $v_i \in V$. If we set $H = F = \bigcup_{\text{node } V_i \in K} \{f_i\}$, then we have $R(D_H)=R(G_K)$. Obviously, in this case the DDBS graph D can be obtained from G in polynomial-time. Hence, if we have a polynomial-time algorithm for computing $R(D_H)$, then we can obtain a polynomial-time algorithm for computing $R(G_K)$ using this construction. However, [4] showed that the problem of computing *KTR* in general is *NP*-hard, so computing DPR in general is *NP*-hard, and the theorem follows.

Theorem 2. Computing DPR for a DDBS with a star topology is *NP*-hard even when each $|FA_i|=2$, where FA_i is the set of databases available at node i.

Proof. We reduce the #LC problem to our problem. For a given network $G=(V_1, L_1)$, where $L_1=\{l_1, l_2, ..., l_n\}$, we construct a DDBS $D = (V_2, L_2, F)$ with a star topology, where $V_2=\{s, v_1, v_2, ..., v_n\}$, $L_2=\{ (s, v_i) \mid 1 \leqq i \leqq n\}$, and $F = \{f_i \mid$ for each vertex $i \in G\}$. Let $FAv_i = \{f_u, f_v \mid$ if $l_i = (u, v) \in G\}$ for $1 \leqq i \leqq n$, $FA_s = \varnothing$ and $H = F$. In the DDBS now define a file spanning tree (FST), which is a tree whose vertics hold all databases $\in H$, i.e., $H \subseteq \bigcup_{\forall \text{ node } i \in FST} \{FA_i\}$. From the construction of D, it is easy to show that there is a one-to-one correspondence between one of the sets of link covers and one FST. The DPR of D, $R(D_H)$, can be expressed as

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

If we set each $p_i = \frac{1}{2}$ for all $1 \leqq i \leqq n$, then we have

$$R(D_H) = \sum_{\text{for all FST } t \in D} (\frac{1}{2})^n$$

$$R(D_H)\, 2^n = \sum_{\text{for all FST } t \in D} 1$$

$= \#$ of FSTs in D

$= \#$ of link covers in G.

Thus, a polynomial-time algorithm for computing $R(D_H)$ over a DDBS with a star topology and each $|FA_i| = 2$ would imply that there is an efficient algorithm for the #LC problem. Since the #LC problem is *NP*-hard, however, Theorem 2 follows.

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

Proof. We reduce the #VC problem to our problem. For a given $G=(V_1, L_1)$, where $|L_1|=n$ and $V_1=\{v_1, v_2, ..., v_m\}$, we construct a DDBS $D=(V_2, L_2, F)$ with a star topology, where $V_2 = V_1 \cup \{s\}$, $L_2=\{l_i=(s, v_i) \mid 1 \leqq i \leqq m\}$, and $F=\{f_i \mid$ for all links $i \in G\}$. Let $FA_i = \{f_j \mid$ for all links 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 database in D and there is a one-to-one corre$\boxed{\text{sp}}$ondence between one of the sets of vertex covers and one FST of D. The DPR of D, $R(D_H)$, can be expressed as

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

If we set each $p_i = \frac{1}{2}$ for all $1 \leqq i \leqq n$, then we have

$$R(D_H) = \sum_{\text{for all FST } t \in D} (\frac{1}{2})^n$$

$$R(D_H)\, 2^n = \sum_{\text{for all FST } t \in D} 1$$

$= \#$ of FSTs in D

$= \#$ of vertex covers in G.

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

Theorem 4. Computing DPR for a DDBS with a 2-tree topology in general is *NP*-hard.

Proof. Assume we have a DDBS graph $D=(V, L, F)$ where $V=\{s, v_1, v_2, ..., v_m\}$ and $L=\{(s, v_i) \mid 1 \leqq i \leqq n\}$ with a star topology. We construct from D a DDBS graph $D'=(V, L', H)$. where $L'= L \cup \{ (v_i, v_{i+1}) \mid 1 \leqq i \leqq n-1\}$. It is easy to see that D' is a 2-tree on n+1 vertics. If we stipulate that all added links (v_i, v_{i+1}), $1 \leqq i \leqq n-1$, of D' have a reliability of 0, then we have $R(D_H)=R(D'_H)$ for any given $H \subseteq F$. This implies that if we have polynomial-time algorithms for a DDBS with a 2-tree topology, then we can obtain polynomial-time algorithms for a DDBS with a star topology. By Theorem 2 and 3, since computing DPR over a DDBS with a star topology in general is *NP*-hard, computing DPR over a DDBS with a 2-tree topology is also *NP*-hard. The theorem follows.

Corollary 1. Computing DPR for a planar DDBS is *NP*-hard.

Corollary 2. Computing DPR for a DDBS with a tree topology is *NP*-hard.

Corollary 3. Computing DPR over a series-parallel DDBS is *NP*-hard.

3 Two Polynomial-Time Algorithms for Computing DPR

We also propose two polynomially-solvable cases of the DPR in which the topology of the DDBS is restricted to a linear or a ring structure. Consider a DDBS with a linear structure with ***n*** edges in which an alternating sequence of distinct nodes and edges $(v_0, e_1, v_1, e_2, ..., v_{n-1}, e_n, v_n)$ is given. For $1 \leqq i \leqq n$, let

FST (file spanning tree) be a tree whose nodes hold all needed databases;

I_i be the FST which starts at edge e_i and has the minimal length;

S_i be the event that all edges in I_i are working;

$Q_i \equiv \prod_{\text{all edge } e_j \in I_i} p_j$ be the probability that S_i occurs

E_i be the event that there exists an operating event S_j between edges e_1 and e_i;

g_i be the number of I_j which lie between e_1 and e_i;

It is easy to see that the DPR of a linear DDBS with n edges can be stated as $\Pr(E_n)$. The following theorem provides a recursive method for computing $\Pr(E_n)$.

Theorem 5.

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

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

Proof. Refer to [9].

The computational complexity of Theorem 5 is $\boldsymbol{O(n*|F|)}$, where F is the set of databases distributed in the linear DDBS.

Recursively applying the factoring theorem and Theorem 5, we have an $\boldsymbol{O(n^{2}*|F|)}$ time algorithm for computing the reliability of a DDBS with a ring structure.

References

[1] A. Satyanarayana and J. N. Hagstrom, "A New Algorithm for the Reliability Analysis of Multi-Terminal Networks," *IEEE Trans. on Reliability*, vol. R-30, pp. 325-334, Oct. 1981.

[2] K. K. Aggrawal and S. Rai, "Reliability Evaluation in Computer-Communication Networks", *IEEE Trans. Reliability*, vol. R-30, pp. 32-35, April 1981.

[3] A. P. Grnarov and M. Gerla, "Multi-terminal Reliability Analysis of Distributed Processing System," in *Proc. 1981 Int. Conf. Parallel Processing*, Aug. 1981, pp. 79-86.

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

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

[6] M. Jerrum, "On the complexity of evaluating multivariate polynomials", Ph.D. thesis, Department of Computer Science, University of Edinburgh, 1981.

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

[8] P. 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.

[9] M. S. Lin and D. J. Chen, "Two Polynomial-Time Algorithms for Computing Reliability in a Linear and a Circular Distributed System," *Proceedings of the PDPTA'97*, Las Vegas, Nevada, USA, June 30 - July 3, 1997.