ご7615

Design and Evaluation of Coteries for Distributed Mutual Exclusion

Tatsuhiro Tsuchiya

September 1998

Design and Evaluation of Coteries for Distributed Mutual Exclusion

Tatsuhiro Tsuchiya

September 1998

Dissertation submitted to the Graduate School of Engineering Science of
Osaka University in partial fulfillment of the requirements
for the degree of Doctor of Engineering

Abstract

The *distributed mutual exclusion problem* has been recognized as one of the most fundamental and important problems in distributed computing since it frequently arises in various kinds of applications. This problem is to guarantee that no more than one computing node will simultaneously access a shared resource that requires exclusive access. In the past two decades, a number of mutual exclusion schemes have been proposed. Among them, quorum-based schemes are known to be elegant and especially robust. In order to achieve mutual exclusion, these schemes use a special set of node groups, called a *coterie*. In a coterie, any two node groups have at least one node in common (*intersection property*), and the node groups are called *quorums*. In a quorum-based scheme, a node has to acquire permission from all the nodes in at least one quorum before entering the critical section. In addition, each node is allowed to give its permission to at most one node. Then, by the intersection property, it is guaranteed that at most one node can enter the critical section at a time.

The performance and dependability of a quorum-based scheme is critically dependent on the coterie adopted. For example, when node or link failures occur, the critical section can be entered only if there is a quorum whose nodes remain connected with each other. So coteries that have a large number of quorums are often desirable in terms of dependability.

The problem of constructing and evaluating coteries have been extensively studied, and a number of methods have been developed. However, most of the design and evaluation methods do not take the topology of the system into account. Though there are a few exceptions, their applicability is limited to special kind of topologies. Coteries on networks with general topologies have not been studied sufficiently yet, and little is known about how to construct or evaluate them. In this dissertation, we address these design and evaluation issues related to coteries on general networks.

For design issues, we consider two performance measures. The first measure is *max-delay*, that is, the maximum communication delay needed for delivering a message between a node wishing to enter the critical section and members of a quorum. The second measure we consider is *availability*, which is the probability that the critical section can be entered in spite of failures. Both the problem of finding coteries with minimal max-delay and the problem of finding coteries with maximal availability have been solved only for the case where the system has special kinds of topologies, such as a tree or a ring. On the contrary, for general networks, these problems have been left open.

In this dissertation, we tackle these problems. We first propose a new algorithm to find max-delay optimal coteries for networks with arbitrary topologies, and show that its time complexity is $O(n^3 \log n)$ where n is the number of nodes.

Then, we propose a method based on 0-1 integer linear programming for constructing coteries with optimal availability. This is an extension of an approach previously proposed by Tang and Natarajan. A major weakness of the original approach is that there has been no analytic method for calculating the values of some parameters used for problem formulation in the case where the system has a general topology. To cope with this weakness, we have developed a new algorithm that computes these parameters and succeeded in obtaining optimal coteries for various networks. To the best of our knowledge, this is the first time that optimal coteries are obtained for general networks with unreliable nodes and links.

As for evaluation issues, we propose graph-theoretic methods for evaluating dependability-related measures. Given a coterie, performance-related measures, such as message complexity or maximum communication delay, are easily computed. However, measures related to dependability are very difficult to compute when the network topology is arbitrary and nodes and links may fail. Only Monte-Carlo simulation and exhaustive state enumeration have been used for this purpose so far.

The first method we propose is for evaluating the availability of coteries. We introduce a new notion called a *Minimal Quorum Spanning Tree* (MQST) and develop the analytic evaluation method based on this notion.

In addition, for assessing dependability of *wr-coteries*, which are an extension of coteries, we propose a method for evaluation of *site resiliency*. This evaluation method is

conceptually similar to the method of availability evaluation for coteries. We also demonstrate the use of the evaluation method for optimizing dependability of a well-known replication control scheme called the weighted voting scheme.

This dissertation is organized as follows: The first and second chapters are introductory ones. In Chapter 1, we summarize related progress and topics and describe an outline of the dissertation. In Chapter 2, we describe a basic model of systems and introduce some basic notions related to quorums and coteries.

In Chapters 3 and 4, we discuss problems of constructing optimal coteries. In Chapter 3, we explain the proposed approach for constructing max-delay optimal coteries on general networks. In Chapter 4, we describe the 0-1 integer programming approach for finding optimal coteries that maximize availability.

In Chapter 5, we present evaluation methods for two measures related to dependability. Both methods employ similar graph-theoretic notions to compute the measures effectively. Through running time analysis, we show their feasibility.

In Chapter 6, we propose another failure model, which is an extension of the model assumed in Chapters 4 and 5. We discuss how to adapt the proposed optimization and evaluation techniques to the new failure model.

Finally, in Chapter 7, we conclude this dissertation with a summary and directions for future work.

List of Major Publication

(1) Tatsuhiro Tsuchiya, Chang Chen, Yoshiaki Kakuda, and Tohru Kikuno, "Calculating performability measures of reconfigurable real-time multiprocessor systems," *Collection of Working Papers of the 1st Workshop on Composability of Fault-Resilient Real-Time Systems* (Dec. 1994).

(2) Tatsuhiro Tsuchiya, Chang Chen, Yoshiaki Kakuda, and Tohru Kikuno, "Calculating performability measures of responsive systems," *Proc. of the 2nd ISSAT International Conference Reliability & Quality in Design*, pp.226-230 (Mar. 1995).

(3) Tatsuhiro Tsuchiya, Yoshiaki Kakuda, and Tohru Kikuno, "Fault-tolerant scheduling algorithm for distributed real-time systems," *Proc. of the 3rd Workshop on Parallel and Distributed Real-Time Systems*, pp.99-103 (Apr. 1995).

(4) Tatsuhiro Tsuchiya, Yoshiaki Kakuda, and Tohru Kikuno, "Modeling and evaluation of responsive multiprocessor systems," *IEICE Transactions on Information and Systems,* vol.J78-D-I, no.8, pp.699-707 (Aug. 1995) (in Japanese). (An English translation appeared in *Systems and Computers in Japan*, vol.27, no.14, pp.20-28 (Dec. 1996).)

(5) Tatsuhiro Tsuchiya, Yoshiaki Kakuda, and Tohru Kikuno, "A new fault-tolerant scheduling technique for real-time multiprocessor systems," *Proc. of the 2nd International Workshop on Real-Time Computing Systems and Applications*, pp.197-202 (Oct. 1995).

(6) Koji Hashimoto, Tatsuhiro Tsuchiya, Yoshiaki Kakuda, and Tohru Kikuno, "Realizing fault-tolerant scheduling using replicated tasks in multiprocessor systems," *Proc.*

of International Workshop on Dependability in Advanced Computing Paradigms, pp.87-92 (June 1996).

(7) Tatsuhiro Tsuchiya, Yoshiaki Kakuda, and Tohru Kikuno, "Three-mode failure model for reliability analysis of distributed programs," *IEICE Transactions on Information and Systems,* vol. E80-D, no. 1, pp.3-9(Jan. 1997).

(8) Eun Mi Kim, Shinji Kusumoto, Tatsuhiro Tsuchiya, and Tohru Kikuno, "An approach to safety verification of object-oriented design specification for an elevator control system," *Proc. of the 3rd International Workshop on Object-Oriented Real-Time Dependable Systems*, pp.256-263 (Feb. 1997).

(9) Koji Hashimoto, Tatsuhiro Tsuchiya, and Tohru Kikuno, "A new approach to realizing fault-tolerant multiprocessor scheduling by exploiting implicit redundancy," *Proc. of the 27th International Symposium on Fault-Tolerant Computing*, pp.174-183 (June 1997).

(10) Tatsuhiro Tsuchiya, Hirofumi Terada, Shinji Kusumoto, Tohru Kikuno, and Eun Mi Kim, "Derivation of safety requirements for safety analysis of object-oriented design documents," *Proc. of the 21th International Computer Software and Application Conference*, pp.312-315 (Aug. 1997).

(11) Tatsuhiro Tsuchiya, Tetsuya Osada, and Tohru Kikuno, "A new heuristic algorithm based on GAs for multiprocessor scheduling with task duplication," *Proc. of the 3rd International Conference on Algorithms and Architectures for Parallel Processing*, pp.295-308 (Dec. 1997).

(12) Eun Hye Choi, Tatsuhiro Tsuchiya, and Tohru Kikuno, "Hierarchical modeling and evaluation of distributed systems," *Proc. of IEEE Pacific Rim International Conference on Fault-Tolerant Systems*, pp.91-96 (Dec. 1997).

(13) Tatsuhiro Tsuchiya and Tohru Kikuno, "Maximizing availability of quorum-based exclusion mechanisms," *Digest of FastAbstracts: FTCS-28 (28th International Symposium on Fault-Tolerant Computing)*, pp.92-93 (June 1998).

(14) Eun Hye Choi, Tatsuhiro Tsuchiya, and Tohru Kikuno, "Availability evaluation of k-coteries on distributed systems with unreliable nodes and links," *Proc. of 1998 International Technical Conference on Circuits/Systems, Computers and Communications*, pp.1685-1690 (July 1998).

(15) Tatsuhiro Tsuchiya, Tetsuya Osada, and Tohru Kikuno, "Genetics-based multiprocessor scheduling using task duplication," *Microprocessors and Microsystems* (to appear).

(16) Tatsuhiro Tsuchiya and Tohru Kikuno, "Dependability evaluation of the weighted voting scheme in the presence of node and link failures," *International Journal of Computer Systems : Science & Engineering* (to appear).

(17) Koji Hashimoto, Tatsuhiro Tsuchiya, and Tohru Kikuno, "A multiprocessor scheduling algorithm for low overhead fault-tolerance," *Proc. of 17th Symposium on Reliable Distributed Systems* (to appear).

(18) Koji Hashimoto, Tatsuhiro Tsuchiya, and Tohru Kikuno, "A new approach to fault-tolerant scheduling using task duplication in multiprocessor systems," *International Journal of Systems and Software* (to appear).

(19) Tatsuhiro Tsuchiya, Masatoshi Yamaguchi, and Tohru Kikuno, "Minimizing the maximum delay for reaching consensus in quorum-based mutual exclusion schemes," *IEEE Transactions on Parallel and Distributed Systems* (submitted).

(20) Eun Hye Choi, Tatsuhiro Tsuchiya, and Tohru Kikuno, "A hierarchical approach to dependability evaluation of distributed systems with replicated resources," *IEEE Transactions on Computers* (submitted).

Acknowledgements

During the course of this work, I have been fortunate to have received assistance from many individuals. I would especially like to thank my supervisor Professor Tohru Kikuno for his continuous support, encouragement, and guidance for this work.

I am also very grateful to the members of my thesis review committee: Professor Hideo Miyahara and Professor Toshinobu Kashiwabara for their invaluable comments and helpful criticisms of this thesis.

Many of the courses that I have taken during my graduate career have been helpful in preparing this thesis. I would like to acknowledge the guidance of Professors Toru Fujiwara, Masaharu Imai, Nobuki Tokura, Ken-ichi Taniguchi, Ken-ichi Hagihara, Mamoru Fujii, Katsuro Inoue, and Akihiro Hashimoto.

I also would like to express my gratitude to Professor Ichiro Suzuki of the University of Wisconsin-Milwaukee for his careful reading of a draft of this thesis. His insightful comments were really valuable for completing this dissertation.

Thanks are also due to many friends in the Department of Informatics and Mathematical Science at Osaka University who gave me many useful comments.

Contents

Chapter 1

Introduction

1.1 Background

Recent advances in computer network technology have led to the increasing popularity of distributed computing systems. A distributed computing system is a system consisting of a collection of autonomous computing nodes connected by a communication network. The nodes typically do not share memory, and communication is solely by means of message passing.

Practical motivations for distributed systems include higher performance or throughput, increased reliability, and sharing resources over a geographically dispersed area. However, pursuing these potential advantages has exposed a broad set of new problems.

Among the problems arising in distributed computing, the *distributed mutual exclusion problem* is recognized as one of the most fundamental and important problems because it frequently arises in various kinds of applications. This problem is to guarantee that no more than one computing node will have access to a shared resource simultaneously. The resource could be, for example, a printer or other device that needs exclusive usage. Or it could be a database or other data structure that requires exclusive access in order to avoid interference among the operations by different processes. A process or computing node currently accessing the resource is said to be in a *critical section.*

In the past two decades, a number of mutual exclusion schemes have been proposed. These schemes are classified into two classes[55]. The first class is token-based schemes (e.g., [48, 60]). In a token-based scheme, a token is circulated among the nodes and only the node that owns the token is allowed to enter the critical section. Though the token-based schemes can achieve mutual exclusion with small communication overhead, they have low fault-tolerance because a failure of a node holding the token leads immediately to the failure of the schemes. Though recovering from such a situation is possible by detecting the failure and regenerating a token, a complex protocol is needed for achieving this (e.g., [43]).

The second class is non-token-based schemes. Among the schemes in this class, those employing the notion of *quorums* are known to be superior to others. In a quorum-based scheme, a node can enter the critical section only when it has gained permission from a certain node group, called a quorum, by message-passing. As described below, these schemes are inherently more fault-tolerant, unlike token-based schemes or other types of non-token-based schemes (e.g.,[50]). Moreover, they are especially suitable for controlling access to shared data that are replicated and distributed throughout the system. Specifically, applications include replication control in database systems[2, 16, 22, 32, 49], name servers[42], emulation of shared memory[36], and dissemination of information[65]. In this dissertation, we focus our discussion on the quorum-based schemes.

In order to achieve mutual exclusion, a quorum-based scheme uses a special set of node groups, called a *coterie*[21]. In a coterie, any two node groups have at least one node in common (*intersection property*), and no group is a superset of any other node group (*minimality property*). A quorum means a node group in a coterie. Once a coterie is determined, mutual exclusion can be achieved as follows: Before entering the critical section, a node has to acquire permission from all the nodes in at least one quorum by communication. In addition, each node is allowed to give its permission to at most one node. Then, by the intersection property, it is guaranteed that at most one node can enter the critical section at a time.

The performance of a quorum-based scheme depends critically on the coterie adopted. For example, if quorums are composed of a small number of nodes, the number of messages a node needs to send for executing the critical section also becomes small. Dependability also depends strongly on the coterie. Even when node or link failures occur, the critical section can still be entered if there is a quorum whose nodes remain connected with each other. Roughly speaking, therefore, coteries that have a large number of quorums are often desirable in terms of dependability.

The problem of constructing coteries with desirable characteristics has been extensively studied and a number of schemes have been developed. One of the schemes uses *voting* to specify quorums[22, 63]. In voting, each node is assigned some votes, and any subset of nodes that has more than half of the system's total votes is considered as a node group containing a quorum. Extensions of voting can be found in [4] and [61]. Most of the other schemes exploit logical structures to construct coteries, such as trees[2], grids[3, 16, 45], hierarchical systems[32, 49], and finite projective planes[37]. A geometric approach is proposed in [35].

Coteries generated by these schemes have different advantages and disadvantages of their own. Generally speaking, coteries with high dependability have low performance (e.g. large message-complexity) and vice versa. Therefore, an appropriate coterie should be chosen according to the requirement of the application. Like coterie construction schemes, various methods have been proposed for evaluating coteries (see [5] for a recent survey). They include attempts to evaluate coteries by capturing the trade-off between dependability and performance (e.g., [33, 40, 52]).

However, most of the design and evaluation methods do not take the topology of the system into account. Though there are a few exceptions, the applicability of such methods is limited to special kinds of topology. Coteries in networks with general topology have not been studied sufficiently yet, and little is known about how to construct or evaluate them. This dissertation addresses these design and evaluation issues related to coteries in general networks.

1.2 Main Results

1.2.1 Design of Coteries

For design issues, we consider two measures and propose methods for constructing coteries that optimize these measures. The first measure is *max-delay*, that is, the maximum communication delay needed for delivering a message between a node wishing to enter the critical section and the members of a quorum[19]. In [19], Fu proposed a delay model and showed that the communication delay depends critically on the coterie. She also proved that polynomial algorithms to find coteries with optimal max-delay exist if the topology of the network is a tree or a ring. However, finding max-delay optimal coteries in general networks has been left as an open problem.

In this dissertation, we solve this problem. We propose an algorithm to find max-delay optimal coteries for networks with arbitrary topology, and show that its time complexity is $O(n^3 \log n)$ where n is the number of nodes.

The second measure we consider is *availability* in the presence of node and link failures. Availability is defined as the probability that the critical section can be entered in spite of failures. The problem to find optimal coteries that maximize availability has been extensively studied for the past decade. For some classes of networks, methods for finding optimal coteries have been proposed[18, 29, 59, 62, 64]. For general networks, however, coteries with optimal availability are very difficult to construct, and only some properties have been investigated recently from a graph-theoretical aspect[23].

To overcome this problem, we propose a new method by extending a 0-1 integer-programming approach proposed by Tang and Natarajan [62]. In its original form, this approach has two main weaknesses. First, for networks with arbitrary topology, there has been no analytic method for calculating the values of some parameters used for the problem formulation. Specifically, this approach requires that for every subset of nodes the probability that it forms an isolated node group due to failures is known. However, no feasible method has existed for computing this probability. The second weakness is that the number of inequality constraints in the formulated problem becomes so large that it cannot be handled practically when the system size grows.

In order to cope with the shortcomings, we propose a new algorithm that computes the probability that each node group becomes an isolated node group. Furthermore, we show that the constraints in the formulated integer-programming problem can be drastically reduced when the network graph has a small number of edges. By means of these results, we have succeeded in obtaining optimal coteries in various networks. To the best of our knowledge, this is the first time that optimal coteries are obtained for general networks with unreliable nodes and links.

1.2.2 Evaluation of Coteries

Coteries that are optimal in some specific measures may have disadvantage in terms of other factors. This means that optimal coteries may not always be adopted. Since an appropriate coterie should be chosen according to the requirement of the application, studying the evaluation of coteries is important as well as developing methods for constructing optimal coteries.

Given a coterie, performance-related measures, such as message complexity or maximum communication delay, are easily computed. For example, since message complexity is known to be proportional to the size of a quorum, it can be computed even without considering the network topology.

On the other hand, measures related to dependability are very difficult to compute when the network topology is arbitrary and nodes and links may fail. Only Monte-Carlo simulation and exhaustive state enumeration have been used for this purpose so far.

In this dissertation, we propose a graph-theoretic method for evaluating the availability of coteries analytically. Though this measure is the most common measure for assessing the dependability of coteries, most of the analytical evaluation methods proposed previously assume partition-free networks and consider neither the topology of the networks nor link failures (see [5] for a recent survey). To overcome this problem, we introduce a new notion called a *Minimal Quorum Spanning Tree* (MQST) and develop an analytic evaluation method based on this notion.

In addition, for evaluating *wr-coteries*, which are an extension of coteries, we propose a method for assessing *site resiliency*[49]. Site resiliency is an important measure for evaluating schemes for controlling replicated data in distributed database systems. This evaluation method is conceptually similar to the method of availability evaluation for coteries. We also demonstrate the use of the evaluation scheme for optimizing the dependability of a well-known replication control scheme called the weighted voting scheme[22].

1.3 Overview of the Dissertation

The rest of this dissertation is organized as follows: In Chapter 2, we describe a basic model of distributed systems and introduce some basic notions related to quorums and coteries.

In Chapter 3, we explain the proposed approach for constructing max-delay optimal coteries in general networks.

In Chapter 4, we describe the 0-1 integer programming approach for finding optimal coteries that maximize availability. In this chapter, we first present a brief explanation of the work by Tang and Natarajan[62], and clarify their shortcomings. Then we show a new algorithm to compute the probability that each node group becomes an isolated node group due to failures, and discuss how to reduce the inequity constraints in the formulated integer-programming problem. By using these proposed techniques, we obtained optimal coteries for various networks. We show the result of this experiment.

In Chapter 5, we present the methods of evaluating the dependability-related measures. All of these methods employ similar graph-theoretic notions to assess the measures more efficiently than exhaustive state enumeration. Through running time analysis, we show their feasibility.

In Chapter 6, we propose another failure model, which is an extension of the model assumed in Chapters 4 and 5. We explain how to adapt the optimization and evaluation techniques to the new failure model.

Finally, we conclude this dissertation with a summary and directions for future work in Chapter 7.

Chapter 2

Basic Concepts

2.1 System Model and Definitions

We consider a computer network modeled by an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is a set of vertices and E is a set of edges incident on vertices in V. Each vertex $v_i \in V$ represents a computing element. We refer to the computing elements as *computing nodes* (nodes, for short). Though this suggests that they are pieces of hardware, it is also possible to think of them instead as logical software processes. We use the letter n to denote $|V|$, the number of nodes in the network. Each edge $e_{i,j} \in E$ represents a communication link (a link, for short) between node v_i and node v_j. We use the term *component* to indicate a node or a link. Without loss of generality, we assume that G is connected and has no self-loops and no parallel edges.

2.2 Coteries

Now we formally define a coterie. The notion of coterie was first introduced by Garcia-Molina and Barbara in [21].

Definition 1 (coterie and quorum) A coterie $\mathcal{C}$ under the set of all nodes V is a set of nonempty subsets of V such that both of the following two conditions hold:*

(i) Intersection property: For any $Q, Q'(\neq Q) \in \mathcal{C}$, Q and Q' have at least one node in common, that is, $Q \cap Q' \neq \emptyset$.

*When V is understood, we will drop it from the discussion.

(ii) Minimality property: For any $Q \in \mathcal{C}$, there is no other element Q' in $\mathcal{C}$ such that $Q' \subset Q$.

Each element in a coterie is called a *quorum* of the coterie.

In the following, we will refer to a nonempty subset of V as a *node group* to avoid confusion.

Example 1 Let the set of all nodes V be $\{v_1, v_2, v_3, v_4\}$. Then consider the following sets of node groups, $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ and $\mathcal{C}_4$.

$$\begin{aligned}
\mathcal{C}_1 &= \{\{v_1\}\} \\
\mathcal{C}_2 &= \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}\} \\
\mathcal{C}_3 &= \{\{v_1, v_2\}, \{v_3, v_4\}\} \\
\mathcal{C}_4 &= \{\{v_1\}, \{v_1, v_2, v_3\}\}
\end{aligned}$$

Among the four sets of node groups, $\mathcal{C}_1$ and $\mathcal{C}_2$ are coteries. $\mathcal{C}_3$ is not a coterie because it does not satisfy the intersection property. $\mathcal{C}_4$ is not a coterie either because it does not satisfy the minimality property.

A coterie is used to achieve mutual exclusion as follows: Before entering the critical section, a node is asked to obtain permission from every nodes in at least one quorum in the coterie. Each node is allowed to give permission to at most one node. If the node has obtained permission from every node in a quorum, it enters the critical section and holds the permission until it leaves the critical section. Because of the intersection property, it is always guaranteed that no two nodes can be in the critical section simultaneously even in the presence of nodes and link failures. (For detailed description of quorum-based mutual exclusion algorithms, see [11, 37, 53].)

2.3 Write-Read Coterie for Replication Control

In distributed database systems, data are usually replicated and distributed throughout the nodes of the systems in a redundant manner. One of the most important advantages of replication is that it masks and tolerates nodes and links failures. Such systems are generally called *replicated database systems*[4, 26].

In replicated databases systems, the replication should be transparent to the users. In order to achieve this, some replication control scheme is required for ensuring that operations performed on a logical data item are reflected on the physical replicas while maintaining their consistency[17]. The replication control scheme must also ensure that the system always presents the most current state of the data even under node and link failures.

For replication control, the notion of a coterie is extended to *write-read coteries*[28].

Definition 2 (wr-coterie) A write-read coterie (wr-coterie, for short) $(\mathcal{W}, \mathcal{R})$ under the set of all nodes V is an ordered pair of two sets of nonempty subsets of V such that all of the following four conditions hold:

(i) For any $Q \in \mathcal{R}, Q' \in \mathcal{W}$, Q and Q' have at least one node in common, that is, $Q \cap Q' \neq \emptyset$.

(ii) For any $Q, Q'(\neq Q) \in \mathcal{W}$, Q and Q' have at least one node in common, that is, $Q \cap Q' \neq \emptyset$.

(iii) For any $Q \in \mathcal{R}$, there is no other element Q' in $\mathcal{R}$ such that $Q' \subset Q$.

(vi) For any $Q \in \mathcal{W}$, there is no other element Q' in $\mathcal{W}$ such that $Q' \subset Q$.

Given a wr-coterie $(\mathcal{W}, \mathcal{R})$, each element in $\mathcal{W}$ is called a *write quorum*, while each element in $\mathcal{R}$ is called a *read quorum*. $\mathcal{W}$ and $\mathcal{R}$ are called the *write quorum set* and the *read quorum set*, respectively.

Exploiting these properties of a wr-coterie, a quorum-based replication control scheme achieves one-copy transparency as follows: A write or read operation can be allowed to proceed to completion if it can get permission from all nodes of a write quorum or a read quorum, respectively. Then the properties of a wr-coterie ensure not only synchronization of conflicting operations but also presenting the single copy view to the uses of the system. Specifically, in a replicated data system, concurrent read and write operations and concurrent write and write operations are the conflicting combinations of operations that have to be synchronized.

For the combination of read and write operations, the condition (i) ensures that the two operations cannot be processed concurrently. This also ensures that read operations will return the latest value written by the last write operation. Likewise, the condition (ii) ensures that write operations are not executed concurrently.

Obviously, if $\mathcal{C}$ is a coterie, $(\mathcal{C}, \mathcal{C})$ is a wr-coterie. This means that a coterie can also be used for replication control as is.

Example 2 Let the set of all nodes V be $\{v_1, v_2, v_3, v_4\}$. The following pairs of sets of node groups, $(\mathcal{W}_1, \mathcal{R}_1)$, $(\mathcal{W}_2, \mathcal{R}_2)$, and $(\mathcal{W}_3, \mathcal{R}_3)$ are examples of wr-coteries.

$$\begin{aligned}
(\mathcal{W}_1, \mathcal{R}_1) &= (\{\{v_1, v_2, v_3, v_4\}\}, \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}\}) \\
(\mathcal{W}_2, \mathcal{R}_2) &= (\{\{v_1, v_2, v_3\}, \{v_1, v_2, v_4\}, \{v_1, v_3, v_4\}, \{v_2, v_3, v_4\}\}, \\
&\quad \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\}) \\
(\mathcal{W}_3, \mathcal{R}_3) &= (\{\{v_1, v_2\}, \{v_2, v_3\}, \{v_1, v_3\}\}, \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_1, v_3\}\})
\end{aligned}$$

Chapter 3

Minimizing Communication Delay

3.1 Introduction

Besides availability and communication complexity, the communication delay needed for achieving mutual exclusion is also being recognized as an important factor for evaluating coteries[19].

In a quorum-based scheme, since a node wishing to enter the critical section has to exchange messages with multiple nodes in the network, communication delay can be a bottleneck in the response time. Precisely, the response time is defined as the length of time that a requesting node spends after it sends request messages and before it exists its critical section[56]. Obviously, it depends on the delay needed for delivering messages between the requesting node and all members of a quorum, and in the algorithm in [37], for example, the response time is equal to $2D + C$, where D is the delay and C is the critical section execution time[56]. In the following, we refer to the delay D as the communication delay (or the delay, for short), and discuss its minimization.

Recently, the notions of *max-delay* and *mean-delay* of coteries have been introduced by Fu [19]. The max-delay of a coterie is the maximum of the delays among all nodes, while the mean-delay is the average. She has shown that there must be a delay-optimal coterie in a special subset of coteries, called *nondominated* (ND) coteries. Based on this result, she has proposed polynomial-time algorithms to find max-delay optimal and mean-delay optimal coteries for systems with special topology, such as trees and rings. Since the number of ND coteries in such networks is very small, the algorithms can efficiently find delay optimal coteries by enumerating ND coteries.

However, finding delay-optimal coteries on general graphs has been left as an open problem. The difficulty of this problem is mainly due to the fact that enumerative approaches are impractical for networks with general topology. This is because the number of ND coteries explodes when the number of nodes exceeds only five [21]. Recently, Bioch and Ibaraki developed an enumeration method based on Boolean algebra [10]. According to [10], however, the method was practically feasible only when the network has less than eight nodes.

In this chapter, we consider the problem of finding max-delay optimal coteries in networks with arbitrary topology. We propose two algorithms to solve this problem. These algorithms take a completely different approach from Fu's in the sense that they do not use the notion of ND coteries. The first algorithm we present finds a max-delay optimal coterie in an arbitrary network with time complexity $O(n^3 \log n)$, where n is equal to the number of nodes. The second algorithm is a modification of the first algorithm. By incorporating heuristics, this algorithm finds a max-delay optimal coterie with smaller mean-delay than the original one. The time complexity of the second algorithm is also $O(n^4)$.

The rest of this chapter is organized as follows: In the next section, we describes the delay model proposed by Fu[19]. In 3.3, we propose an algorithm to find max-delay optimal coteries. In 3.4, we extend the proposed algorithm in order to generate max-delay optimal coteries with smaller mean-delay. In 3.5, we show the results obtained by applying the proposed algorithms to several sample networks.

3.2 A Delay Model

We adopt the model used in [19] and describe it in this section.

3.2.1 Virtual Distance

We assume that each edge $e \in E$ of the network graph G is assigned a positive real number $w(e)(> 0)$ as its weight. Figure 3.1 illustrates an example of a computer network. The number attached to each edge represents its weight.

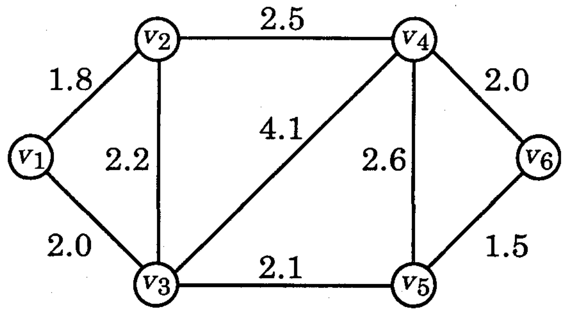

Figure 3.1: An example of a network.

For two distinct nodes v_i and v_j, the virtual distance $dist(v_i, v_j)(= dist(v_j, v_i))$ between the two nodes is defined as the length of the shortest path on G, where the length of a path is the sum of the weights assigned to the edges in the path. The virtual distance represents the communication delay between the two nodes. In Figure 3.1, for example, $dist(v_1, v_6)$ is equal to 5.6. We assume that for any node v_i, the virtual distance $dist(v_i, v_i)$ is equal to zero.

3.2.2 Delay of Coteries

Adopting definitions from [19], we introduce the notions of the delay of a node in a coterie and the max-delay and the mean-delay of a coterie. The delay of node v_i in coterie $\mathcal{C}$, or $delay(v_i, \mathcal{C})$, is given by

$$delay(v_i, \mathcal{C}) = \min_{Q \in \mathcal{C}}\{\max_{v \in Q}\{dist(v_i, v)\}\},$$

and the max-delay and the mean-delay of coterie $\mathcal{C}$ are given by

$$\begin{aligned} max\text{-}delay(\mathcal{C}) &= \max_{v \in V}\{delay(v, \mathcal{C})\} \\ mean\text{-}delay(\mathcal{C}) &= \frac{1}{|V|}\sum_{v \in V} delay(v, \mathcal{C}) \end{aligned}$$

Intuitively, the delay of a node in a coterie means the communication delay for achieving consensus when the node accesses its nearest quorum. The max-delay of a coterie is the maximum of the delays among all nodes, and the mean-delay of a coterie is the arithmetic mean of the delays of all nodes.

Example 3 Consider a network shown in Figure 3.1. Let $\mathcal{C} = \{\{v_2, v_4\}(= Q_1), \{v_2, v_5\}(= Q_2), \{v_4, v_5\}(= Q_3)\}$ be a coterie in the network. Now take a node v_1 as an example. Among the three quorums, Q_2 is the nearest quorum from v_1, and $delay(v_1, \mathcal{C})$ is equal to $\max_{v \in Q_2}\{dist(v_1, v)\} = dist(v_1, v_5) = 4.1$. For v_2, v_3, v_4, v_5, v_6, such nearest quorums are Q_1, Q_2, Q_1, Q_3, Q_3, respectively. Since $delay(v_1, \mathcal{C}) = 4.1$, $delay(v_2, \mathcal{C}) = 2.5$, $delay(v_3, \mathcal{C}) = 2.2$, $delay(v_4, \mathcal{C}) = 2.5$, $delay(v_5, \mathcal{C}) = 2.6$, and $delay(v_6, \mathcal{C}) = 2.0$, $max\text{-}delay(\mathcal{C})$ and $mean\text{-}delay(\mathcal{C})$ are equal to 4.1 and 2.65, respectively.

Then, a max-delay optimal coterie is defined as follows:

Definition 3 (Max-delay optimal coterie) Let $\mathcal{UC}$ be the set of all coteries. A coterie $\mathcal{C} \in \mathcal{UC}$ is said to be max-delay optimal if and only if

$$max\text{-}delay(\mathcal{C}) = \min_{\mathcal{C}' \in \mathcal{UC}}\{max\text{-}delay(\mathcal{C}')\}.$$

3.2.3 Delay of Write-Read Coteries

Given a wr-coterie $\mathcal{B} = (\mathcal{W}, \mathcal{R})$, the write-delay and the read-delay for node v_i is given by

$$write\text{-}delay(v_i, \mathcal{B}) = \min_{Q \in \mathcal{W}}\{\max_{v \in Q}\{dist(v_i, v)\}\},$$

$$read\text{-}delay(v_i, \mathcal{B}) = \min_{Q \in \mathcal{R}}\{\max_{v \in Q}\{dist(v_i, v)\}\}.$$

The delay of node v_i in V and the max-delay of wr-coterie $\mathcal{B}$ are given by

$$\begin{aligned} delay(v_i, \mathcal{B}) &= \max\{write\text{-}delay(v_i, \mathcal{B}), read\text{-}delay(v_i, \mathcal{B})\} \\ max\text{-}delay(\mathcal{B}) &= \max_{v \in V}\{delay(v, \mathcal{B})\} \end{aligned}$$

Lemma 1 (A.Fu [19]) If $\mathcal{C}$ is a max-delay optimal coterie, $(\mathcal{C}, \mathcal{C})$ is a wr-coterie with smallest max-delay.

This lemma allows us to reduce the problem of finding a max-delay optimal wr-coterie to that of finding a max-delay optimal coterie. In the following, therefore, we focus our discussion on finding max-delay optimal coteries.

3.3 Finding Max-Delay Optimal Coteries

As mentioned before, the problem we consider in this chapter is to find a max-delay optimal coterie for given $G = (V, E)$ and w.

Suppose that virtual distance $dist(v_i, v_j)$ for any $v_i, v_j \in V$ has been computed from G and w. In fact, this is possible in polynomial time by using some previously proposed algorithms (e.g., [41]). Then, we can restate the problem by explicitly specifying a quorum from which each node gets consensus.

Problem 1 Given V and virtual distance $dist(v_i, v_j)$ for any $v_i, v_j \in V$, find an n-tuple of node groups $(N_1, N_2, \cdots, N_n)$ that minimizes $\max_{v_i \in V}\{\max_{v \in N_i}\{dist(v_i, v)\}\}$, under the conditions that (i) for any two groups N_i and N_j, $N_i \cap N_j \neq \emptyset$, and (ii) for any two groups N_i and N_j, $N_i \not\subset N_j$.

Suppose $(N_1, N_2, \cdots, N_n)$ is an optimal solution to Problem 1, and let $\mathcal{C}$ be the set of all N_i's $(i = 1, 2, \cdots, n)$. Note that the number of elements in $\mathcal{C}$ is not necessarily n since N_i may be equal to N_j for some $i, j (\neq i)$. Then, $\mathcal{C}$ is obviously a max-delay optimal coterie.

In order to solve this problem, we consider a more tractable problem, removing Condition (ii) from Problem 1.

Problem 2 Given V and virtual distance $dist(v_i, v_j)$ for any $v_i, v_j \in V$, find an n-tuple of node groups $(N_1, N_2, \cdots, N_n)$ that minimizes $\max_{v_i \in V}\{\max_{v \in N_i}\{dist(v_i, v)\}\}$, under the condition that for any two node groups N_i and N_j, $N_i \cap N_j \neq \emptyset$.

Lemma 2 Suppose that $(N_1, N_2, \cdots, N_n)$ is an optimal solution to Problem 2. Let $\mathcal{C}$ be the set of N_i's such that any other $N_j (i \neq j)$ is not a proper subset of N_i, i.e., $\mathcal{C} = \{N_i | N_i \in \mathcal{N}$ such that $\forall N_j \in \mathcal{N} - \{N_i\}, N_j \not\subset N_i\}$, where $\mathcal{N}$ is the set of all N_i's $(i = 1, 2, \cdots, n)$. (Note that the number of elements in $\mathcal{N}$ is not necessarily n since N_i may be equal to N_j for some $i, j (\neq i)$.) Then, $\mathcal{C}$ is a max-delay optimal coterie.

Proof: By definition, it is clear that $\mathcal{C}$ is a coterie. If N_i is not in $\mathcal{C}$, then there is another node group N_j in $\mathcal{C}$ such that $N_j \subset N_i$. Now consider another n-tuple $(M_1, M_2, \cdots, M_n)$ such that if $N_i \in \mathcal{C}$, then $M_i = N_i$, otherwise M_i is equal to a node group N_j in $\mathcal{C}$ such that $N_j \subset N_i$. Then, since $M_i \subseteq N_i$ holds for any i, $\max_{v \in M_i}\{dist(v_i, v)\} \leq \max_{v \in N_i}\{dist(v_i, v)\}$ also holds for any i. It is then clear $\max_{v_i \in V}\{\max_{v \in M_i}\{dist(v_i, v)\}\} \leq \max_{v_i \in V}\{\max_{v \in N_i}\{dist(v_i, v)\}\}$.

In addition, since $(N_1, N_2, \cdots, N_n)$ is an optimal solution to Problem 2, $\max_{v_i \in V}\{\max_{v \in M_i}\{dist(v_i, v)\}\} \geq \max_{v_i \in V}\{\max_{v \in N_i}\{dist(v_i, v)\}\}$. Hence $\max_{v_i \in V}\{\max_{v \in M_i}\{dist(v_i, v)\}\} = \max_{v_i \in V}\{\max_{v \in N_i}\{dist(v_i, v)\}\}$. This implies that $(M_1, M_2, \cdots, M_n)$ is also an optimal solution to Problem 2.

Since Problem 2 differs from Problem 1 only in that it does not impose Condition (ii) on solutions, the inequality $\max_{v_i \in V}\{\max_{v \in M_i}\{dist(v_i, v)\}\} \leq \max_{v_i \in V}\{\max_{v \in L_i}\{dist(v_i, v)\}\}$ holds if $(L_1, L_2, \cdots, L_n)$ is an optimal solution to Problem 1. Notice that $(M_1, M_2, \cdots, M_n)$ satisfies Conditions (i) and (ii) in Problem 1. Hence, it is also an optimal solution to Problem 1. By definition, $\mathcal{C}$ is equal to the set of M_i's, thus $\mathcal{C}$ is a max-delay optimal coterie. □

Now we define $NB_i(r)$ as the set of all nodes whose virtual distance from v_i is equal to or smaller than r, i.e., $NB_i(r) = \{v \in V | dist(v_i, v) \leq r\}$. In addition, let min be the smallest positive real number such that for any $v_i, v_j \in V$, $NB_i(min)$ and $NB_j(min)$ intersect, i.e., $NB_i(min) \cap NB_j(min) \neq \emptyset$. Then the following lemma holds.

Lemma 3 $(NB_1(min), NB_2(min), \cdots, NB_n(min))$ is an optimal solution to Problem 2.

Proof: (By contradiction.) If $(NB_1(min), NB_2(min), \cdots, NB_n(min))$ is not an optimal solution to Problem 2, there exists another n-tuple $(N_1, N_2, \cdots, N_n)$ such that $N_i \cap N_j \neq \emptyset$ for any i, j and $min > \max_{v_i \in V}\{\max_{v \in N_i}\{dist(v_i, v)\}\}$. Let $d = \max_{v_i \in V}\{\max_{v \in N_i}\{dist(v_i, v)\}\}$. Then, since $N_i \subseteq NB_i(d)$ holds for any i, $NB_i(d) \cap NB_j(d) \neq \emptyset$ for any i, j. Because $\max_{v_i \in V}\{\max_{v \in NB_i(d)}\{dist(v_i, v)\}\} = d$, $(NB_1(d), NB_2(d), \cdots, NB_n(d))$ is also an optimal solution to Problem 2. Since $d < min$, this is a contradiction to the definition of min. □

Let $\mathcal{C}_{opt}$ be the set of $NB_i(min)$'s such that any $NB_j(min) (i \neq j)$ is not a proper subset of $NB_i(min)$, i.e., $\mathcal{C}_{opt} = \{NB_i(min) | NB_i(min) \in \mathcal{NB}$ such that $\forall NB_j(min) \in \mathcal{NB} - \{NB_i(min)\}, NB_j(min) \not\subset NB_i(min)\}$, where $\mathcal{NB}$ is the set of all $NB_i(min)$'s $(i = 1, 2, \cdots, n)$. Then, we obtain the following theorem.

Theorem 1 $\mathcal{C}_{opt}$ is a max-delay optimal coterie.

Proof: It is clear from Lemma 1 and Lemma 2. □

Theorem 1 leads to an algorithm for finding a max-delay optimal coterie. Figure 3.2 shows this algorithm. It consists of three steps.

At Step 1, the virtual distance between every pair of nodes is calculated. This can be done, for example, by using Floyd's classical algorithm with time complexity $O(n^3)$ [20]. (There are also faster algorithms such as [41].)

At Step 2, min and $NB_i(min)$'s are calculated. At the beginning of this step, elements of $\bigcup_{v_i,v_j \in V} \{dist(v_i, v_j)\}$, that is, all candidates for min are sorted in ascending order and stored in $a_1, a_2, \cdots$. Using this data structure and Function Intersection_Check, min can be obtained by binary search. Function Intersection_Check checks whether $NB_i(r) \cap NB_j(r) \neq \emptyset$ holds for any i, j or not.

Since the number of candidates for min is at most $\frac{n(n-1)}{2}$, it takes $O(n^2 logn)$ time for sorting them by merge sort or heap sort in the worst case. The **while** loop in Step 2 is iterated $O(\log n^2)$ times. Function Intersection_Check requires $O(n^3)$ time per invocation because the **for** loop in this function is repeated at most $\frac{n(n-1)}{2}$ times and the **if** statement checking whether or not $D_i \cap D_j$ is empty takes $O(n)$ time. Hence it takes $O(n^3 \log n)$ time for completing the **while** loop. Thus the time complexity of Step 2 is $O(n^3 \log n)$.

At Step 3, all supersets are removed from $NB_i(min)$'s. The remaining elements form a max-delay optimal coterie. The time complexity of this step is $O(n^3)$ time because each **if** statement takes $O(n)$ and the **for** loop is iterated $\frac{n(n-1)}{2}$ times. Consequently, it is seen that the time complexity of this algorithm is $O(n^3 \log n)$.

Example 4 Consider the network shown in Figure 3.1. The proposed algorithm works as follows: At Step 1, the virtual distance between every pair of nodes is calculated. Then, the value of min and $NB_i(min)$'s are calculated. First, candidates for min are sorted as follows:

$$1.5,\ 1.8,\ 2.0,\ 2.1,\ 2.2,\ 2.5,\ 2.6,\ 3.6,\ 4.1,\ 4.3,\ 4.5,\ 5.6$$

Using binary search, the value of min is calculated. In this case, $min = 3.6$. Then, each

$NB_i(min)$ is calculated and stored in D_i as follows:

$$\begin{aligned}
D_1 &= \{v_1, v_2, v_3\} \\
D_2 &= \{v_1, v_2, v_3, v_4\} \\
D_3 &= \{v_1, v_2, v_3, v_5, v_6\} \\
D_4 &= \{v_2, v_4, v_5, v_6\} \\
D_5 &= \{v_3, v_4, v_5, v_6\} \\
D_6 &= \{v_3, v_4, v_5, v_6\}
\end{aligned}$$

We can obtain a max-delay optimal coterie by removing all supersets from the $NB_i(min)$'s. In this case, it is $\{\{v_1, v_2, v_3\}, \{v_2, v_4, v_5, v_6\}, \{v_3, v_4, v_5, v_6\}\}$. Obviously, its max-delay is 3.6. The mean-delay of this coterie is 2.533.

3.4 Extension for Reducing Mean-Delay

The algorithm presented in the previous section can successfully find a max-delay optimal coterie $\mathcal{C}_{opt}$ for any given network G. However, there may be other max-delay optimal coteries whose mean delay is smaller than $\mathcal{C}_{opt}$. In order to find such a max-delay optimal coterie with small mean-delay, we propose a new algorithm by modifying the original algorithm.

Specifically, we insert a new step (referred to as Step 2') into the original algorithm between Step 2 and Step 3. At the end of Step 2, $NB_1(min)$, $NB_2(min)$, $\cdots$, $NB_n(min)$ have been obtained. Each $NB_i(min)$ is stored in D_i in the algorithm in Figure 3.2. In Step 2', we examine the nodes in $D_1, D_2, \cdots, D_n$ one by one, in a certain order, and remove those nodes whose removal does not destroy the property that every pair of D_i and D_j intersect. Even with such an additional stage that reduces D_i's, it is guaranteed that the obtained coterie is max-delay optimal and its mean-delay is not larger than $\mathcal{C}_{opt}$. The following theorem ensures this.

Theorem 2 Let $(N_1, N_2, \cdots, N_n)$ be an n-tuple of node groups such that $N_i \cap N_j \neq \emptyset$ for any $v_i, v_j \in V$ and $N_i \subseteq NB_i(min)$ for any $v_i \in V$. Let $\mathcal{C}$ be the set of N_i's such that no other $N_j (i \neq j)$ is a proper subset of N_i, i.e., $\mathcal{C} = \{N_i | N_i \in \mathcal{N}$ such that $\forall N_j \in \mathcal{N} - \{N_i\}, N_j \not\subset N_i\}$, where $\mathcal{N}$ is the set of all N_i's $(i = 1, 2, \cdots, n)$. Then, $\mathcal{C}$ is a max-delay optimal coterie and *mean-delay*$(\mathcal{C}) \leq$ *mean-delay*$(\mathcal{C}_{opt})$.

```
Step 1: /* Calculate the virtual distance between every pair of nodes */
  Execute an algorithm such as [20] or [41].

Step 2: /* Find $min$ by binary search */
  Sort $\bigcup_{v_i, v_j \in V} \{dist(v_i, v_j)\}$ in ascending order and store in $a_1, a_2, ..., a_m$
  $il := 1$, $iu := m$
  While $(il \neq iu)$ do
    $i := \lfloor (il + iu)/2 \rfloor$
    If Intersection_Check($a_i$) = true then $iu := i$ else $il := i + 1$
  END_While
  $D_i := \{v \in V | dist(v_i, v) \leq a_{il}\}$ for all $v_i \in V$ /* $min = a_{il}, D_i = NB_i(min)$ */

Step 3: /* Remove all supersets from $NB_i(min)$'s. */
  $tmp$ := a list of $D_i$'s
  For all pair $v_i, v_j \in V$ do
    If $D_i \subseteq D_j$ remove $D_j$ from $tmp$
    If $D_j \subset D_i$ remove $D_i$ from $tmp$
  END_For
  $C_{opt}$ := a set of node groups in $tmp$ /* max-delay optimal coterie */
END

Function Intersection_Check($r$)
  $D_i := \{v \in V | dist(v_i, v) \leq r\}$ for all $v_i \in V$
  For all pair $v_i, v_j \in V$ do
    If $D_i \cap D_j = \emptyset$ then return(false)
  END_For
  return(true)
END_Function
```

Figure 3.2: Proposed algorithm for obtaining a max-delay optimal coterie.

Proof: Since $N_i \subseteq NB_i(min)$ for any i, $\max_{v \in N_i}\{dist(v_i, v)\} \leq \max_{v \in NB_i(min)}\{dist(v_i, v)\}$ holds for any i. By Lemma 2, $(NB_1(min), NB_2(min), \cdots, NB_n(min))$ is an optimal solution to Problem 2. Hence $\max_{v_i \in V}\{\max_{v \in N_i}\{dist(v_i, v)\}\} = min$ and $(N_1, N_2, \cdots, N_n)$ is also an optimal solution to Problem 2. By Lemma 1, it is then clear that $\mathcal{C}$ is a max-delay optimal coterie. By definition, for any quorum $Q \in \mathcal{C}_{opt}$, there is $Q' \in \mathcal{C}$ such that $Q' \subseteq Q$. Note that if $Q' \subseteq Q$, then for any node v_i $\max_{v_j \in Q'}\{dist(v_i, v_j)\} \leq \max_{v_j \in Q}\{dist(v_i, v_j)\}$. Hence $mean\text{-}delay(\mathcal{C}) \leq mean\text{-}delay(\mathcal{C}_{opt})$. □

In order to determine which nodes are checked and removed, we do the following: Let a tuple (D_i, v_j) denote a node v_j in D_i. (So there are $\sum_{v_i \in V} |D_i|$ tuples that have to be considered.) We choose (D_i, v_j) with the largest value of $dist(v_i, v_j)$ first, since this is intuitively natural. If there are more than one tuple with the largest value, a node is chosen from D_i such that $|D_i|$ is the largest. Figure 3.3 shows Step 2' of the modified algorithm. (Steps 1, 2 and 3 are omitted since they are exactly the same as the original algorithm shown in Figure 3.2.)

Since the total number of tuples considered is less than n^2, the **while** loop in Step 2' is repeated no more than n^2 times. Likewise, one tuple can be selected in $O(n^2)$ time at an iteration of the **while** loop. Checking the intersection property also requires $O(n^2)$ time. Hence the time complexity of this step is $O(n^4)$. Since the original algorithm, which is the one without Step 2', is of $O(n^3 \log n)$ time complexity, that of the modified algorithm is $O(n^4)$.

Example 5 Consider the network shown in Figure 3.1. Since Steps 1 and 2 of the modified algorithm are the same as the original algorithm, each D_i in the modified algorithm has been set to $NB_i(min)$ at the end of Step 2, as shown in the previous example. Thus there are 24 tuples that have to be considered in this case.

Step 2' reduces D_i's as follows: First, among the 24 tuples, tuple (D_i, v_j) such that the value of $dist(v_i, v_j)$ is the largest is selected. In this case, (D_3, v_6) and (D_6, v_3) have the greatest value $3.6 (= dist(v_3, v_6) = dist(v_6, v_3))$. Since $|D_3| (= 5)$ is larger than $|D_6| (= 4)$, (D_3, v_6) is chosen first. Because $D_3 - \{v_6\}$, i.e., $\{v_1, v_2, v_3, v_5\}$ has at least one common node with each of D_1, D_2, D_4, D_5, and D_6, v_6 is removed from D_3. Then D_i's become as follows:

```
Step 2': /* Reduce $D_i$'s */
 $POOL := \bigcup_{D_i;1\leq i\leq n} \{(D_i, v_j)|v_j \in D_i\}$
 While $POOL \neq \emptyset$ do
   /* Select one tuple $(D_i, v_j)$ */
   From $POOL$, choose $(D_i, v_j)$ such that
     $dist(v_i, v_j) = \max_{(D_i,v_j)\in POOL}\{dist(v_i, v_j)\}$.
   (If the number of such tuples is more than one,
   choose a tuple with the largest value of $|D_i|$ from them.)
   $POOL := POOL - (D_i, v_j)$
   /* Check whether $(D_i, v_j)$ can be removed or not */
   $D_i := D_i - \{v_j\}$
   For all $D_k$ with $k \neq i$
     If $D_i \cap D_k = \emptyset$
       $D_i := D_i \cup \{v_j\}$, Break the for loop
   END_For
 END_While
```

Figure 3.3: Step 2' of the modified algorithm.

$$
\begin{aligned}
D_1 &= \{v_1, v_2, v_3\} \\
D_2 &= \{v_1, v_2, v_3, v_4\} \\
D_3 &= \{v_1, v_2, v_3, v_5\} \\
D_4 &= \{v_2, v_4, v_5, v_6\} \\
D_5 &= \{v_3, v_4, v_5, v_6\} \\
D_6 &= \{v_3, v_4, v_5, v_6\}
\end{aligned}
$$

Next, (D_6, v_3) is chosen for checking. However, if v_3 is removed from D_6, D_6 no longer intersects with D_1. Thus this node is not removed. (D_4, v_5) is chosen next since $dist(v_4, v_5)$ is the largest among the remaining unchecked tuples. This process is repeated until all tuples have been checked. Consequently, D_i's become as follows:

$$D_1 = \{v_2, v_3\}$$
$$D_2 = \{v_2, v_3\}$$
$$D_3 = \{v_2, v_3\}$$
$$D_4 = \{v_2, v_6\}$$
$$D_5 = \{v_3, v_6\}$$
$$D_6 = \{v_3, v_6\}$$

A max-delay optimal coterie is derived from the D_i's at Step 3. In this case, it is $\{\{v_2, v_3\}, \{v_2, v_6\}, \{v_3, v_6\}\}$. The mean-delay of this coterie is 2.433. (Recall that the mean-delay of the coterie obtained by the original algorithm is 2.533.)

3.5 Experimental Results

Using C language, we coded the original algorithm and the modified algorithm. For Step 1, we adopted Floyd's algorithm [20]. We took a collection of networks from [14] shown in Figure 3.4, and used it as the topology of sample networks. Weights were assigned to the edges randomly. For each network in Figure 3.4, we executed the programs on a SUN Ultra 1 workstation and obtained max-delay coteries. In all cases, the running time needed to generate a max-delay optimal coterie was less than 0.1 second.

Tables 3.1 and 3.2 show max-delay coteries obtained by the two algorithms. Table 3.3 summarizes the max-delays and mean-delays of the coteries. This result clearly shows that the max-delay optimal coteries generated by the modified algorithm have much smaller mean-delays than those generated by the original algorithms. For Network 6, for example, the mean-delay of the coterie generated by the modified algorithm is smaller than that of the coterie generated by the original algorithm by about 35%.

The modified algorithm is an approximation method for solving the problem to find a max-delay optimal coterie such that its mean-delay is minimized. Unfortunately, we have not yet developed an optimal algorithm to solve this problem, and we do not know the complexity of this problem either. Besides, finding mean-delay optimal coteries in arbitrary networks is still left as an open problem.

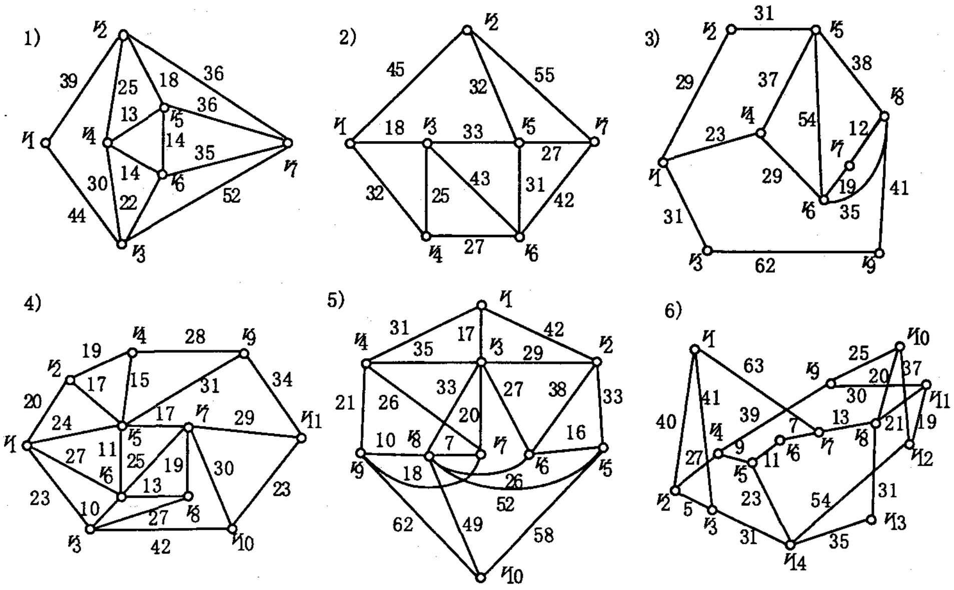

Figure 3.4: Sample networks.

Table 3.1: Optimal coteries generated by the original algorithm.

Network	Max-delay optimal coterie
1	$\{\{v_1, v_2, v_3\}, \{v_2, v_5, v_6, v_7\}, \{v_1, v_3, v_4, v_5, v_6\}, \{v_2, v_3, v_4, v_5, v_6\}\}$
2	$\{\{v_1, v_2, v_5\}, \{v_5, v_6, v_7\}, \{v_1, v_3, v_4, v_6\}\}$
3	$\{\{v_3, v_7, v_8, v_9\}, \{v_1, v_2, v_3, v_4, v_5\}, \{v_1, v_2, v_3, v_4, v_9\}, \{v_1, v_4, v_5, v_6, v_7, v_8\}, \{v_4, v_5, v_6, v_7, v_8, v_9\}\}$
4	$\{\{v_7, v_{10}, v_{11}\}, \{v_4, v_5, v_9, v_{11}\}, \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}\}$
5	$\{\{v_7, v_8, v_{10}\}, \{v_1, v_3, v_4, v_7, v_8, v_9\}, \{v_1, v_2, v_3, v_5, v_6, v_7, v_8\}, \{v_2, v_3, v_5, v_6, v_7, v_8, v_9\}\}$
6	$\{\{v_1, v_2, v_3, v_7\}, \{v_5, v_6, v_7, v_8, v_{10}, v_{11}, v_{13}, v_{14}\}, \{v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{14}\}, \{v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}, \{v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_{12}, v_{13}, v_{14}\}, \{v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{14}\}\},$

Table 3.2: Optimal coteries generated by the modified algorithm.

Network	Max-delay optimal coterie
1	$\{\{v_2,v_3\}, \{v_2,v_4\}, \{v_2,v_6\}, \{v_3,v_4,v_6\}\}$
2	$\{\{v_1,v_5\}, \{v_1,v_6\} \{v_5,v_6\}\}$
3	$\{\{v_3,v_4\}, \{v_4,v_7\}, \{v_4,v_8\}, \{v_3,v_7,v_8\}\}$
4	$\{\{v_5,v_7\}, \{v_5,v_{11}\}, \{v_7,v_{11}\}\}$
5	$\{\{v_2,v_8\}, \{v_3,v_8\}, \{v_7,v_8\}, \{v_2,v_3,v_7\}\}$
6	$\{\{v_2,v_7\}, \{v_4,v_7\}, \{v_6,v_7\}, \{v_7,v_{14}\}, \{v_2,v_4,v_6,v_{14}\}\}$

Table 3.3: Max-delay and mean-delay of obtained coteries.

Network	Max-delay	Mean-delay	
		Original algorithm	Modified algorithm
1	44.0	36.86	28.57
2	51.0	40.86	37.86
3	62.0	58.33	47.78
4	41.0	34.09	28.91
5	56.0	45.90	30.50
6	63.0	55.79	36.71

Chapter 4

Maximizing Availability in the Presence of Failures

4.1 Introduction

In many distributed systems a mutual exclusion mechanism is required to work even when nodes or communication links fail. For example, consider a system that manages replicated data. If failures partition the system into isolated node groups, which we call *partition groups*, we probably do not want users at different partition groups to update the replicated data concurrently since this would make the data inconsistent.

In such a situation, quorum-based mechanisms provide desirable fault-tolerance capability: When network partitioning occurs due to failures, only nodes in the partition group that includes a quorum are allowed to enter the critical section. (Notice that by definition, there is no more than one partition group that includes a quorum.)

Given for each component the probability that it is operational, the availability of a mutual exclusion mechanism is defined as the probability that the critical section can be entered in spite of failures[8]. This means that the availability is equal to the probability that there is at least one quorum whose nodes are operational and connected. It is then obvious that the availability of the mutual exclusion mechanism depends completely on the structure of the coterie adopted by the mechanism. Thus we use the term *the availability of a coterie* to refer to the availability of the mutual exclusion mechanism based on this coterie.

As stated before, a number of schemes for coterie construction have been developed. Among them, voting[22, 63] is shown to be especially effective for generating highly available coteries. Especially for partition-free systems, such as fully connected networks with perfectly reliable links and ethernet-like systems, Spasojevic and Berman proved in [59] that coteries constructed by voting have optimal availability if votes are assigned to nodes by Tong and Kain's algorithm[64].

However, when network partitioning has to be taken into account, coteries that optimize availability are very difficult to construct. Only for networks with special topologies, such as trees and rings[29], polynomial-time algorithms have been developed based on graph theory.* For coteries in general networks, only some properties have been investigated recently from a graph-theoretical aspect[23].

Apart form these graph-theoretical approaches, Tang and Natarajan proposed an integer-programming approach[62]. They formulated the problem of finding optimal coteries as a sparse 0-1 integer programming problem. Using this approach, they obtained optimal coteries for five example networks.

Since this approach does not rely on any special topologies of networks, we expect that it is the only viable approach for designing optimal coteries in general networks. However, this approach has two main weaknesses. First, it requires the probability that each node group becomes a partition group for formulating the problem, but no analytic method for calculating the probability has been proposed. In fact, they computed the probability by case-by-case analysis, assuming that links of the five networks are failure-free to make the computation easier.

The second weakness is its scalability. In the integer programming problem, the number of variables and that of inequality constraints grow exponentially as the number of nodes increases. Hence when the number of nodes exceeds 10, the original approach is no longer feasible.

The purpose of our study is to overcome these problems: As for the first problem, we newly develop an algorithm that computes the probability that each node group forms a partition group. Furthermore, for the second problem, we show that the constraints in the formulated integer-programming problem can be drastically reduced when the network

*Assuming that links are reliable and nodes have the same reliability, Diks et al. proposed a method for constructing optimal coteries on complete multipartite graphs[18].

graph has a small number of edges.

By combining these results with the work in [62], we have succeeded in obtaining optimal coteries for various networks. To the best of our knowledge, this is the first time that optimal coteries have been obtained for general networks with unreliable nodes and links.

The rest of this chapter is organized as follows: In 4.2, we describe a failure model that we assume. In 4.3, we explain the approach by Tang and Natarajan[62], and clarify its weaknesses. In 4.4, we show the new algorithm for computing the probability that each partition group is formed. In 4.5, we show how to reduce the inequality constraints in the integer programming problem. In 4.6, we present the result of an experiment we conducted. In the experiment, we obtained optimal coteries in various networks by the proposed method. In addition, we conduct comparison study of the optimal coteries and heuristically-designed coteries.

4.2 Model and Definitions

4.2.1 A Failure Model

We assume that each component may fail and has two states: operational and failed. Failures of components are mutually independent and they are fail-stop failures. We assume that for every component the probability that it is operational is given a priori. Let $p_i (0 < p_i \leq 1)$ and $p_{i,j} (0 < p_i \leq 1)$ denote the probability that v_i is operational and the probability that $e_{i,j}$ is operational, respectively. We refer to these probabilities as the *reliabilities* of components.

Based on these assumption, we introduce some definitions in the following. For $VC \subseteq V$ and $EC \subseteq E$, we call a tuple (VC, EC) a *configuration* of G. Note that a configuration may not be a graph because it may have an edge incident on a node that is not included in the configuration. The *current configuration* is defined as a configuration (VC, EC) such that VC and EC are the set of all currently operational nodes and the set of all currently operational edges. In other words, the current configuration represents the current status of the network. The universe of all possible configurations UC is defined as follows:

$$UC = \{(VC, EC) | VC \subseteq V \text{ and } EC \subseteq E\}$$

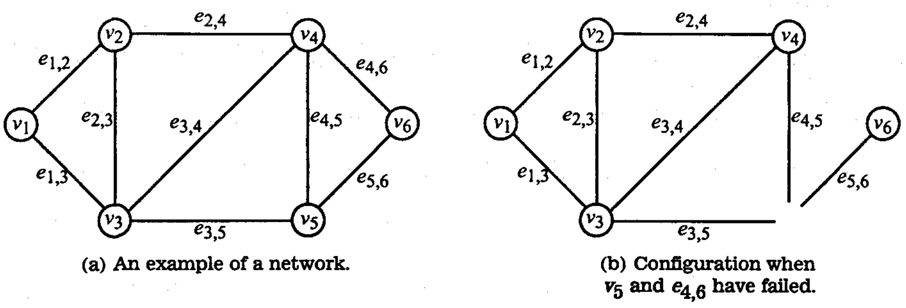

Figure 4.1: An example of a network.

Nodes in a configuration may be divided into some isolated groups called *partition groups.*

Definition 4 (partition group) Given a configuration $(VC, EC) \in UC$, a *partition group* in (VC, EC) is defined as a subset of V, say $N(\subseteq V)$, such that all of the following conditions hold.

(i) $N \subseteq VC$.

(ii) All nodes in N are connected by links in EC.

(iii) For any of N's proper supersets, say N', either $N' \not\subseteq VC$, or not all nodes in N' are connected by links in EC.

We use notation $\mathcal{P}(VC, EC)$ to represent the set of all partition groups appearing in a configuration (VC, EC).

Example 6 Consider a network $G = (V, E)$ shown in Figure 4.1(a). When a node v_5 and an edge $e_{4,6}$ have failed, the current configuration becomes $(\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}, e_{3,5}, e_{4,5}, e_{5,6}\})$. Figure 4.1(b) schematically illustrates the current configuration (say (VC, EC)). (Note that a configuration may not be a subgraph of G.) All partition groups appearing in (VC, EC) are $\{v_1, v_2, v_3, v_4\}$ and $\{v_6\}$, that is, $\mathcal{P}(VC, EC) = \{\{v_1, v_2, v_3, v_4\}, \{v_6\}\}$.

4.2.2 Availability of Coterie

In the presence of node and link failures, the critical section can be entered if and only if there is a partition group that contains a quorum in the current configuration. Therefore, the availability of the mutual exclusion mechanism, i.e., the availability of the coterie on which the mechanism works, is equal to the probability that there is such a partition group in the current configuration. In this subsection, we present the formal definition of the availability of a coterie $\mathcal{C}$ in a network $G = (V, E)$.

Given a coterie $\mathcal{C}$, let $UC_{\mathcal{C}}(\subseteq UC)$ be the set of all configurations in which there is a partition group that contains a quorum in $\mathcal{C}$, that is,

$$UC_{\mathcal{C}} = \{(VC, EC) \in UC \mid \text{there exist } Q(\in \mathcal{C}) \text{ and } N(\in \mathcal{P}(VC, EC)) \text{ such that } Q \subseteq N\}.$$

For example, consider a coterie $\mathcal{C} = \{\{v_3, v_4\}, \{v_3, v_6\}, \{v_4, v_6\}\}$ on the network shown in Figure 4.1(a). In this case, $UC_{\mathcal{C}}$ includes the configuration (VC, EC) illustrated in Figure 4.1(b) because there are partition group $N = \{v_1, v_2, v_3, v_4\}(\in \mathcal{P}(VC, EC))$ and quorum $Q = \{v_3, v_4\}(\in \mathcal{C})$ such that $Q \subseteq N$. In other words, the critical section can still be entered if and only if the current configuration belongs to $UC_{\mathcal{C}}$.

Definition 5 (availability) The availability of a coterie $\mathcal{C}$ is the probability that there is a partition group that contains a quorum of the coterie $\mathcal{C}$ in the current configuration, i.e.,

$$\text{Availability} = \sum_{(VC,EC) \in UC_{\mathcal{C}}} prob(VC, EC),$$

where $prob(VC, EC)$ is the probability that the current configuration is (VC, EC).

The value of $prob(VC, EC)$ can be obtained as follows: Let u_j and $u_{j,k}$ be Boolean variables that represent the event that a node v_j is operational and the event that a link $e_{j,k}$ is operational, respectively. Then, the event that the current configuration is equal to (VC, EC) is represented by a Boolean product

$$\bigwedge_{v_i \in VC} u_i \wedge \bigwedge_{v_i \in V-VC} \overline{u}_i \wedge \bigwedge_{e_{i,j} \in EC} u_{i,j} \wedge \bigwedge_{e_{i,j} \in E-EC} \overline{u}_{i,j}.$$

Since failures of components are assumed to be independent, and the probability that the event corresponding to u_i (or $u_{i,j}$) occurs is equal to p_i (or $p_{i,j}$), $prob(VC, EC)$ is given by

$$prob(VC, EC) = \prod_{v_i \in VC} p_i \prod_{v_i \in V-VC} (1 - p_i) \prod_{e_{i,j} \in EC} p_{i,j} \prod_{e_{i,j} \in E-EC} (1 - p_{i,j}).$$

In this chapter, we say a coterie $\mathcal{C}$ under V is *optimal*, if its availability is higher than or equal to any other coteries under V.

4.3 An Integer-Programing Approach

Tang and Natarajan proposed an integer-programming approach for obtaining optimal coteries[62]. Since this approach is not specialized in topologies of networks, we expect that it is the only reasonable way to design optimal coteries in general networks. In this section, we explain this approach and clarify its shortcomings.

4.3.1 Acceptance Set

Precisely, Tang and Natarajan proposed a method for finding an optimal *acceptance set*, instead of an optimal coterie.

Definition 6 (acceptance set) An acceptance set $\mathcal{S}$ under the set of all nodes V is a set of nonempty subsets of V such that the following condition holds.

(i) Intersection property: If Q and $Q'(\neq Q)$ are in $\mathcal{S}$, then Q and Q' have at least one node in common, that is, $Q \cap Q' \neq \emptyset$.

Thus an acceptance set differs from a coterie only in that the minimality property is not required for an acceptance set. Thus, for any acceptance set $\mathcal{S}$, a set of minimal node groups in $\mathcal{S}$ comprises a coterie.

Let $h(N)$ be the probability that a node group N forms a partition group in the current configuration, i.e.,

$$h(N) = \sum_{\{(VC,EC)\in UC \mid N\in\mathcal{P}(VC,EC)\}} prob(VC,EC).$$

An acceptance set $\mathcal{S}$ is said to be optimal if $\sum_{N\in\mathcal{S}} h(N)$ is the largest among all possible acceptance sets. Since the notion of an acceptance set is very similar to that of a coterie, we can easily show that an optimal coterie is obtained directly from a given optimal acceptance set.

Lemma 4 If an acceptance set $\mathcal{S}$ is optimal, then a set of node groups $\mathcal{C} = \{N \in \mathcal{S} | \forall N' \in \mathcal{S}, N' \not\subset N\}$ is a coterie with the maximum availability.

Proof: (by contradiction) Given an optimal acceptance set $\mathcal{S}$, another acceptance set $\mathcal{S}' = \{N | \exists N' \in \mathcal{S}, N' \subseteq N \subseteq V\}$ is determined uniquely. Since $\mathcal{S} \subseteq \mathcal{S}'$, $\mathcal{S}'$ is also

optimal. By the definition of $\mathcal{S}'$, $\{N \in \mathcal{S}' | \forall N' \in \mathcal{S}', N' \not\subset N\}$ is equal to $\mathcal{C}$. Now suppose $\mathcal{C}$ is not optimal, and there is another coterie $\mathcal{C}'(\neq \mathcal{C})$ with higher availability than $\mathcal{C}$. Consider an acceptance set $\mathcal{S}'' = \{N | \exists N' \in \mathcal{C}', N' \subseteq N \subseteq V\}$. By the definition of $h(N)$, the availability of $\mathcal{C}$ (or $\mathcal{C}'$) is $\sum_{\{N|\exists Q \in \mathcal{C}, Q \subseteq N \subseteq V\}} h(N)$ (or $\sum_{\{N|\exists Q \in \mathcal{C}', Q \subseteq N \subseteq V\}} h(N)$). Hence, the availability of $\mathcal{C}$ is equal to $\sum_{N \in \mathcal{S}'} h(N)$, and the availability of $\mathcal{C}'$ is $\sum_{N \in \mathcal{S}''} h(N)$. Then $\sum_{N \in \mathcal{S}'} h(N) < \sum_{N \in \mathcal{S}''} h(N)$. Since this is a contradiction to the optimality of $\mathcal{S}'$, $\mathcal{C}$ is optimal.
□

This lemma indicates that if an acceptance set $\mathcal{S}$ that maximizes $\sum_{N \in \mathcal{S}} h(N)$ is found, an optimal coterie can be obtained directly from it by removing all nonminimal elements.

4.3.2 Formulation

The problem of finding an optimal acceptance set can be formulated as a 0-1 integer programming problem as follows:

First, we define a *partition* of V as a set of disjoint node groups such that any node in V appears in exactly one of the node groups. Since the total number of node groups is $2^n - 1$, each partition of V can be represented by a 0-1 row vector of length $2^n - 1$, where the ith element is a 1 if and only if the ith node group exists in that partition.[†] Let $\boldsymbol{F}(\mathcal{R})$ denote this (row) vector corresponding to a partition $\mathcal{R}$. Additionally, let $\boldsymbol{H}$ be a row vector of length $2^n - 1$, where the ith element is a real number equal to the probability that the ith node group becomes a partition group, i.e., $h(N)$ where N denotes the ith node group.

Now let $\boldsymbol{A}$ be a 0-1 column vector of length $2^n - 1$. $\boldsymbol{A}$ can specify any set of node groups uniquely in such a way that a 1 in the ith position of $\boldsymbol{A}$ represents that the ith node group is an element of this set, while a 0 means that it is not. If $\boldsymbol{A}$ specifies an acceptance set $\mathcal{S}$, then it is clear that $\boldsymbol{HA} = \sum_{N \in \mathcal{S}} h(N)$ holds. Tang et al. have proven that the necessary and sufficient condition that the set corresponding to $\boldsymbol{A}$ is an acceptance set is that $\boldsymbol{F}(\mathcal{R})\boldsymbol{A} \leq 1$ holds for every partition $\mathcal{R}$. Consequently, the problem of finding an

[†]In principle, any encoding scheme that establishes a one to one map between the set of all node groups and the set of integers $\{1, 2, \cdots, 2^n - 1\}$ is adequate for this purpose. In this chapter, we encode a given node group $N \subseteq V$ as an integer i in such a way that the binary representation of i is equal to $b_n b_{n-1} \cdots b_1$ where b_j is 1 iff v_j is included in N.

acceptance set $\mathcal{S}$ that maximizes $\sum_{N \in \mathcal{S}} h(N)$ is formulated into a 0-1 integer-programming problem shown below:

Maximize $\boldsymbol{HA}$ subject to the constraint $\boldsymbol{F}(\mathcal{R})\boldsymbol{A} \leq 1$ for every $\mathcal{R}$.

Here elements in $\boldsymbol{A}$ are variables to be optimized. Commercial packages are available for solving the integer-programming problem. If an optimal solution of $\boldsymbol{A}$ is obtained, its corresponding acceptance set is the one we want to find.

Example 7 Consider a network $G = (V, E)$, where $V = \{v_1, v_2, v_3\}$ and $E = \{e_{1,2}, e_{1,3}\}$. Concerning reliabilities of components, assume $p_1 = 0.7, p_2 = 0.8, p_3 = 0.9$ and $p_{1,2} = p_{1,3} = 0.9$. In this case, there are seven different node groups. Suppose that they are ordered as follows:

$$\{v_1\}, \{v_2\}, \{v_1, v_2\}, \{v_3\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_1, v_2, v_3\}.$$

All partitions of V are

$$\{\{v_1\}, \{v_2, v_3\}\}, \{\{v_2\}, \{v_1, v_3\}\}, \{\{v_3\}, \{v_1, v_2\}\}, \{\{v_1\}, \{v_2\}, \{v_3\}\}, \{\{v_1, v_2, v_3\}\}.$$

Any partition $\mathcal{R}$ can be represented by a 0-1 row vector $\boldsymbol{F}(\mathcal{R})$ in length seven. For example, given a partition $\mathcal{R} = \{\{v_1\}, \{v_2, v_3\}\}$, its corresponding vector is

$$\boldsymbol{F}(\mathcal{R}) = [1\ 0\ 0\ 0\ 0\ 1\ 0].$$

Vector $\boldsymbol{H}$ has seven elements, each of which is for one node group, and each element is equal to $h(N)$, that is, the probability that its corresponding node group N appears as a partition group. For the system, $\boldsymbol{H}$ is

$$\begin{aligned} \boldsymbol{H} &= [h(\{v_1\})\ h(\{v_2\})\ h(\{v_1, v_2\})\ h(\{v_3\})\ h(\{v_1, v_3\})\ h(\{v_2, v_3\})\ h(\{v_1, v_2, v_3\})] \\ &= [0.037240\ \ 0.296000\ \ 0.095760\ \ 0.333000\ \ 0.158760\ \ 0.000000\ \ 0.408240] \end{aligned}$$

For example, $\{v_1\}$ becomes a partition group when both of the following two conditions hold, provided that v_1 is operational. The first condition is that $e_{1,2}$ or v_2 has failed, while the second one is that $e_{1,3}$ or v_3 has failed. So $h(\{v_1\})$ is $0.7 \times ((1 - 0.9) + (1 - 0.8) - (1 - 0.9)(1 - 0.8)) \times ((1 - 0.9) + (1 - 0.9) - (1 - 0.9)(1 - 0.9)) = 0.7 \times 0.28 \times 0.19 = 0.03724$.

Let $\boldsymbol{A}$ be a 0-1 column vector of length 7. Then the problem of finding an optimal acceptance set is formulated into the following integer programming problem.

$$\begin{aligned}
&\text{Maximize } \boldsymbol{HA} \text{ subject to}\\
&[1\ 0\ 0\ 0\ 0\ 1\ 0\ ]\boldsymbol{A} \le 1,\\
&[0\ 1\ 0\ 0\ 1\ 0\ 0\ ]\boldsymbol{A} \le 1,\\
&[0\ 0\ 1\ 1\ 0\ 0\ 0\ ]\boldsymbol{A} \le 1,\\
&[1\ 1\ 0\ 1\ 0\ 0\ 0\ ]\boldsymbol{A} \le 1,\\
&[0\ 0\ 0\ 0\ 0\ 0\ 1\ ]\boldsymbol{A} \le 1.
\end{aligned}$$

Some commercial linear-programming packages are available for solving the integer programming problem. The following are the optimal value of the objective function $\boldsymbol{HA}$, and one of the optimal solutions of $\boldsymbol{A}$.

$$\boldsymbol{HA} = 0.90, \quad \text{and} \quad \boldsymbol{A}^T = [0\ 0\ 0\ 1\ 1\ 1\ 1].$$

This result means that the optimal availability is equal to 0.90 and that an acceptance set $\mathcal{S}$ that maximizes $\sum_{N \in \mathcal{S}} h(N)$ is $\{\{v_3\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_1, v_2, v_3\}\}$. Consequently, we obtain the following optimal coterie $\mathcal{C}$ by removing all nonminimal elements in $\mathcal{S}$. In this case,

$$\mathcal{C} = \{\{v_3\}\}.$$

4.3.3 Weaknesses

Thus this approach can obtain optimal coteries in networks with arbitrary topology if for every node group N, the probability that it becomes a partition group, that is, $h(N)$, is known. In the above example, since the system has only three nodes, this probability can be obtained by case-by-case analysis. However, it is obvious that such an ad hoc approach is not feasible for larger systems. (Notice that the total number of node groups is $2^n - 1$.) Unfortunately, no analytical method for computing such a probability for general networks has been developed. For example, Venkaiah and Jalote used this probability to solve another problem[66], and they obtained it by simulation approximately.

Table 4.1: The size of the formulated 0-1 integer programming problem.

# of nodes	# of variables	# of inequality constraints
6	63	203
7	127	877
8	255	4,140
9	511	21,147
10	1,023	115,975
11	2,047	678,570
12	4,095	4,213,597

Another problem of this approach is its scalability. As is already shown in [62], the number of variables and that of constraints in the 0-1 integer programming problem formulated by this approach grow exponentially in the number of nodes of networks. (Note that each variable corresponds to a node group, while each constraint corresponds to a partition of V.) Table 4.1 shows the size of the 0-1 integer programming problem for networks of up to 12 nodes. The number of variables is equal to that of node groups, i.e., $2^n - 1$. Here, the number of inequality constrains was computed using an algorithm presented in [62].

From this table, it is found that especially the number of constraints in the problem grows very rapidly when the number of nodes increases. Therefore, the large number of the constraints becomes one of the main factors that limit the tractable size of the problem.

In the following sections, we show how to overcome the shortcomings. First, we propose a new algorithm for computing the value of $h(N)$ for every node group N. Since the total number of node groups is already $2^n - 1$, the problem of computing this probability for every node group is computationally intractable. In 4.4, therefore, we first explain the algorithm and then show the feasibility of the algorithm for moderate sized networks.

Moreover, in 4.5, we show that it is possible to reduce the numbers of variables and inequality constraints in many situations. Roughly speaking, we show that every node group that does not form a partition group can be ignored in the problem formulation if a certain condition holds. Unless G is a complete graph, such node groups always exist. In 4.6, we show when the network graph has a small number of edges, the numbers of variables and inequality constraints can be reduced drastically.

4.4 Computing $h(N)$'s

A straightforward method for computing $h(N)$'s for all N's is to examine all possible configurations.[‡] Specifically, all $h(N)$'s can be obtained by for every configuration $(VC, EC) \in UC$, determining all partition groups in (VC, EC) and computing $prob(VC, EC)$. However, since the total number of possible configurations, that is, $|UC|$, is equal to $2^{|V|+|E|}$, this method is clearly impractical.

The key idea behind the proposed algorithm for computing all $h(N)$'s more efficiently is to examine subgraphs of G, instead of configurations themselves. A subgraph of G is a special configuration such that for every edge in the configuration, both of its end nodes are also in the configuration. Since the total number of possible subgraphs is much less than that of configurations, this algorithm runs much faster than the straightforward method that examines all configurations.

4.4.1 Basic Analysis

Given a node group $V'(\subseteq V)$, we denote by $E_{V'}$ the set of edges having both of its end nodes in V'. Additionally, given a set of edges $E'(\subseteq E)$, we let E'^c denote $E - E'$, regarding E as the universe of all edges and $E - E'$ as the complement of E'.

For a configuration (VC, EC), the maximal subgraph of G embedded in the configuration is uniquely determined by removing edges such that at least one of their end nodes is not included in VC. That is, this subgraph is equal to (VS, ES) such that $VS = VC$ and $ES = EC \cap E_{VC}$. Now let $Conf(VS, ES)$ be the set of configurations such that their corresponding maximal subgraph is (VS, ES). Then, $Conf(VS, ES)$ is written as

[‡]One might think that k-terminal reliability algorithms (e.g.[54]) can be used in order to compute $h(N)$. (These algorithms compute the probability that all nodes in a given set are connected via operational paths.) However, they cannot be used for this purpose because for computing $h(N)$, not only the connectivity of nodes in N but also the (un)connectivity of every superset of N have to be considered.

follows:

$$Conf(VS, ES) = \{(VC, EC) | VC = VS \text{ and } ES \subseteq EC \subseteq ES \cup E^c_{VS}\}.$$

Example 8 Consider the network shown in Figure 4.1. For example, for the configuration shown in Figure 4.1(b), the maximal subgraph in this configuration is $(VS, ES) = (\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}\})$. On the other hand, since E^c_{VS} is $\{e_{3,5}, e_{4,5}, e_{5,6}\}$, all elements of $Conf(VS, ES)$ are as follows:

$$\begin{gathered}
(\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}\}),\\
(\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}, e_{3.5}\}),\\
(\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}, e_{4.5}\}),\\
(\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}, e_{5.6}\}),\\
(\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}, e_{3.5}, e_{4,5}\}),\\
(\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}, e_{3.5}, e_{5,6}\}),\\
(\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}, e_{4.5}, e_{5,6}\}),\\
(\{v_1, v_2, v_3, v_4, v_6\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}, e_{3.5}, e_{4,5}, e_{5,6}\})
\end{gathered}$$

By definition, $Conf(VS_1, ES_1) \cap Conf(VS_2, ES_2) = \emptyset$ for any two different subgraphs (VS_1, ES_1) and (VS_2, ES_2), and $UC = \bigcup_{(VS,ES) \in US} Conf(VS, ES)$ where US is the universe of all subgraphs of G. In addition, partition groups appearing in a configuration are exactly the same as those appearing in its corresponding maximal subgraph. Hence, the probability that a node group N becomes a partition group is written as follows:

$$\begin{aligned}
h(N) &= \sum_{\{(VS,ES) \in US | N \in \mathcal{P}(VS,ES)\}} \sum_{(VC,EC) \in Conf(VS,ES)} prob(VC, EC)\\
&= \sum_{\{(VS,ES) \in US | N \in \mathcal{P}(VS,ES)\}} \hat{prob}(VS, ES),
\end{aligned}$$

where $\hat{prob}(VS, ES) = \sum_{(VC,EC) \in Conf(VS,ES)} prob(VC, EC)$. In other words, $\hat{prob}(VS, ES)$ is the probability that the corresponding maximal subgraph of the current configuration is equal to (VS, ES).

Due to the above equation, it is found that if $\hat{prob}(VS, ES)$ can be obtained easily for a given subgraph (VS, ES), then $h(N)$ can be computed by examining only subgraphs of G, instead of all configurations.

In the following, we show how to calculate $\hat{prob}(VS, ES)$. The event that a configuration (VC, EC) belonging to $Conf(VS, ES)$ becomes the current configuration is represented by the following Boolean expression

$$\bigwedge_{v_i \in VC} u_i \wedge \bigwedge_{v_i \in V-VC} \overline{u}_i \wedge \bigwedge_{e_{i,j} \in EC} u_{i,j} \wedge \bigwedge_{e_{i,j} \in E-EC} \overline{u}_{i,j}.$$

Now note that E is partitioned into four disjoint subsets $EC \cap E_{VS} (= ES)$, $EC^c \cap E_{VS} (= E_{VS} - ES)$, $EC \cap E^c_{VS}$, and $EC^c \cap E^c_{VS}$. Then, the above expression is transformed as follows:

$$\bigwedge_{v_i \in VS} u_i \wedge \bigwedge_{v_i \in V-VS} \overline{u}_i \wedge \bigwedge_{e_{i,j} \in ES} u_{i,j} \wedge \bigwedge_{e_{i,j} \in E_{VS}-ES} \overline{u}_{i,j} \wedge \bigwedge_{e_{i,j} \in EC \cap E^c_{VS}} u_{i,j} \wedge \bigwedge_{e_{i,j} \in EC^c \cap E^c_{VS}} \overline{u}_{i,j}.$$

In the expression, $\bigwedge_{v_i \in VS} u_i \wedge \bigwedge_{v_i \in V-VS} \overline{u}_i \wedge \bigwedge_{e_{i,j} \in ES} u_{i,j} \wedge \bigwedge_{e_{i,j} \in E_{VS}-ES} \overline{u}_{i,j}$ is independent of EC and can be determined only by (VS, ES). Let $U_{(VS,ES)}$ denote this part. Then, the event that the current configuration belongs to $Conf(VS, ES)$ is represented by

$$\begin{aligned}
&\bigvee_{\{(VC,EC) \in Conf(VS,ES)\}} (U_{(VS,ES)} \wedge \bigwedge_{e_{i,j} \in EC \cap E^c_{VS}} u_{i,j} \wedge \bigwedge_{e_{i,j} \in EC^c \cap E^c_{VS}} \overline{u}_{i,j}) \\
&= U_{(VS,ES)} \wedge \bigvee_{\{(VC,EC) \in Conf(VS,ES)\}} (\bigwedge_{e_{i,j} \in EC \cap E^c_{VS}} u_{i,j} \wedge \bigwedge_{e_{i,j} \in EC^c \cap E^c_{VS}} \overline{u}_{i,j}) \\
&= U_{(VS,ES)} \wedge \bigvee_{\{EC | ES \subseteq EC \subseteq ES \cup E^c_{VS}\}} (\bigwedge_{e_{i,j} \in EC \cap E^c_{VS}} u_{i,j} \wedge \bigwedge_{e_{i,j} \in EC^c \cap E^c_{VS}} \overline{u}_{i,j})
\end{aligned}$$

Since $ES \cap E^c_{VS} = \emptyset$, the following equation holds.

$$\bigvee_{\{EC | ES \subseteq EC \subseteq ES \cup E^c_{VS}\}} (\bigwedge_{e_{i,j} \in EC \cap E^c_{VS}} u_{i,j} \wedge \bigwedge_{e_{i,j} \in EC^c \cap E^c_{VS}} \overline{u}_{i,j}) = \bigvee_{\{\emptyset \subseteq E' \subseteq E^c_{VS}\}} (\bigwedge_{e_{i,j} \in E'} u_{i,j} \wedge \bigwedge_{e_{i,j} \in E^c_{VS}-E'} \overline{u}_{i,j})$$

Clearly, the above expression is tautology, that is, this expression is always satisfied regardless of assignment for $u_{i,j}$'s. Thus, it is found that $U_{(VS,ES)}$ corresponds to the event that the current configuration belongs to $Conf(VS, ES)$. Consequently, we obtain

$$\hat{prob}(VS, ES) = \prod_{v_i \in VS} p_i \prod_{v_i \in V-VS} (1 - p_i) \prod_{e_{i,j} \in ES} p_{i,j} \prod_{e_{i,j} \in E_{VS}-ES} (1 - p_{i,j}).$$

4.4.2 Algorithm for Computing $h(N)$'s

Based on the above equation, we develop an algorithm for computing the probability that each node group becomes a partition group. For every subgraph (VS, ES), the proposed algorithm determines all partition groups in (VS, ES) and then accumulates $\hat{prob}(VS, ES)$ for the partition groups. Figure 4.2 shows a formal description of this algorithm. In this algorithm, all partition groups appearing in a subgraph are determined incrementally and stored in a set Set_PGs at substep 2.1. Then, $\hat{prob}(VS, ES)$ is computed and accumulated at substep 2.2.

```
/* Step 1: Initialization */
  For (every node group $N$) do
    $h(N) := 0$
  END_For
/* Step 2: Accumulation of $\hat{prob}(VS, ES)$ */
  For (every subgraph $(VS, ES)$) do
    /* Substep 2.1 Determination of Partition Groups in $(VS, ES)$ */
      $V' := VS, Set\_PGs := \emptyset$
      While ($N'$ is not empty) do
        Select one node $v_i$ from $N'$
        $PG := \{v_i\}$, $N' := N - \{v_i\}$
        $AN := \{v_j \in N' | v_j$ is adjacent to $v_i\}$
        While ($AN$ is not empty) do
          $PG := PG + AN$, $N' := N' - AN$
          $AN := \{v_j \in N' | v_j$ is adjacent to a node in $PG\}$
        END_While
        $Set\_PGs := Set\_PGs + PG$
      END_While
    /* Substep 2.2 Computation and Accumulation of $\hat{prob}(VS, ES)$ */
      $E_{VS} := \{e_{i,j} \in E|$ both $v_i$ and $v_j$ are in $VS\}$
      Compute $\hat{prob}$
      /* $\hat{prob} := \prod_{v_i \in VS} p_i \prod_{v_i \in V-VS} (1 - p_i) \prod_{e_{i,j} \in ES} p_{i,j} \prod_{e_{i,j} \in EA_{VS}-ES} (1 - p_{i,j})$ */
      For (every node group $N$ in $Set\_PGs$) do
        $h(N) := h(N) + \hat{prob}$
      END_For
  END_For
```

Figure 4.2: Algorithm for computing $h(N)$'s

4.5 Reduction of Variables and Inequality Constrains

In this section, we show how to reduce the numbers of variables and inequality constraints in the integer programming problem. In Tong and Natarajan's approach, the number of variables is equal to $2^n - 1$, and the number of inequality constraints is the same as that of all partitions of V. As described above, these numbers grow very rapidly as the network size grows, and this becomes a major bottleneck to the feasibility of the approach.

We show that if there are node groups that never form partition groups (i.e., node groups $N1, N2, \cdots$, such that $h(N1) = h(N2) = \cdots = 0$), the problem is transformed into a problem with fewer variables and inequality constraints. Such node groups always exist unless the network graph G is a complete graph. Let $\mathcal{K}$ denote the set of all of these node groups, i.e., $\mathcal{K} = \{N | N \subseteq V, N \neq \emptyset, h(N) = 0\}$.

It is easy to show that variables corresponding to these node groups are not necessary for formulating the problem. Note that if $h(N1) = 0$, for any acceptance set $\mathcal{S}$, $\sum\limits_{N \in \mathcal{S}} h(N) = \sum\limits_{N \in \mathcal{S} - \{N1\}} h(N)$. Therefore we can only consider acceptance sets not including such node groups. This means the optimization problem is formulated as

Maximize $\boldsymbol{H'A'}$ subject to the constraint $\boldsymbol{F'}(\mathcal{R})\boldsymbol{A'} \leq 1$ for every $\mathcal{R}$.

where $\boldsymbol{H'}$, $\boldsymbol{A'}$, and $\boldsymbol{F'}(\mathcal{R})$ are vectors of length $2^n - 1 - |\mathcal{K}|$ that are obtained by removing elements corresponding to the node groups in $\mathcal{K}$ from $\boldsymbol{H}$, $\boldsymbol{A}$, and, $\boldsymbol{F}(\mathcal{R})$, respectively.

At this point, the number of inequalities in the constraint part of this problem is the same as the number of all partitions of V. Next we show that for any partition $\mathcal{R}$ including node groups in $\mathcal{K}$, the inequality constraint $\boldsymbol{F'}(\mathcal{R})\boldsymbol{A'} \leq 1$ can be omitted if $h(\{v_i\}) > 0$ holds for every $v_i \in V$. More formally, we show that if $h(\{v_i\}) > 0$ holds for every $v_i \in V$,

$\boldsymbol{F'}(\mathcal{R})\boldsymbol{A'} \leq 1$ for every $\mathcal{R}$

is equivalent to

$\boldsymbol{F'}(\mathcal{R})\boldsymbol{A'} \leq 1$ for every $\mathcal{R}$ such that $\forall N \in \mathcal{R}, h(N) \neq 0$.

We can prove this by showing that for any partition $\mathcal{R}$ including a node group in $\mathcal{K}$, there exists another partition $\mathcal{R}'$ such that 1) $N \notin \mathcal{K}$ for every $N \in \mathcal{R}'$, and 2) $\boldsymbol{F'}(\mathcal{R}')\boldsymbol{A'} > 1$ holds whenever $\boldsymbol{F'}(\mathcal{R})\boldsymbol{A'} > 1$ holds. In fact, given any partition $\mathcal{R}$ of V, $\bigcup\limits_{N \in \mathcal{R} - \mathcal{K}} \{N\} \cup \bigcup\limits_{N \in \mathcal{R} \cap \mathcal{K}} \bigcup\limits_{v \in N} \{\{v\}\}$ becomes such a partition $\mathcal{R}'$ of V if $h(\{v_i\}) \neq 0$ for every $v_i \in V$.

Consequently, if $h(\{v_i\}) \neq 0$ for every $v_i \in V$, the problem of finding an optimal acceptance set is formulated into the following 0-1 integer programming problem.

Maximize $\boldsymbol{H'A'}$ subject to the constraint $\boldsymbol{F'}(\mathcal{R})\boldsymbol{A'} \leq 1$ for every $\mathcal{R}$ such that $\forall N \in \mathcal{R}, h(N) \neq 0$.

Example 9 Consider the network discussed in the previous example. Since $h(\{v_2, v_3\}) = 0$, we consider acceptance sets not including $\{v_2, v_3\}$. Suppose that the possible node groups excluding $\{v_2, v_3\}$ are ordered as follows:

$$\{v_1\}, \{v_2\}, \{v_1, v_2\}, \{v_3\}, \{v_1, v_3\}, \{v_1, v_2, v_3\}.$$

Then $\boldsymbol{H}'$ becomes

$$\begin{aligned} \boldsymbol{H}' &= [h(\{v_1\})\ h(\{v_2\})\ h(\{v_1, v_2\})\ h(\{v_3\})\ h(\{v_1, v_3\})\ h(\{v_1, v_2, v_3\})] \\ &= [0.037240\ \ 0.296000\ \ 0.095760\ \ 0.333000\ \ 0.158760\ \ 0.408240] \end{aligned}$$

All partitions of V such that $\{v_2, v_3\}$ is not included are

$$\{\{v_2\}, \{v_1, v_3\}\}, \{\{v_3\}, \{v_1, v_2\}\}, \{\{v_1\}, \{v_2\}, \{v_3\}\}, \{\{v_1, v_2, v_3\}\}.$$

Any of these partitions can be represented by a 0-1 row vector $\boldsymbol{F}'(\mathcal{R})$ of length six. For example, given a partition $\mathcal{R} = \{\{v_2\}, \{v_1, v_3\}\}$, its corresponding vector is

$$\boldsymbol{F}'(\mathcal{R}) = [0\ 1\ 0\ 0\ 1\ 0].$$

Let $\boldsymbol{A}'$ be a 0-1 column vector of length six. Then the problem of finding an optimal acceptance set is formulated as the following integer programming problem.

$$\begin{aligned} &\text{Maximize } \boldsymbol{H}'\boldsymbol{A}' \text{ subject to} \\ &[0\ 1\ 0\ 0\ 1\ 0\ ]\boldsymbol{A} \leq 1, \\ &[0\ 0\ 1\ 1\ 0\ 0\ ]\boldsymbol{A} \leq 1, \\ &[1\ 1\ 0\ 1\ 0\ 0\ ]\boldsymbol{A} \leq 1, \\ &[0\ 0\ 0\ 0\ 0\ 1\ ]\boldsymbol{A} \leq 1. \end{aligned}$$

(Note that there is no inequality constraint corresponding to a partition $\{\{v_1\}, \{v_2, v_3\}\}$, since if $\boldsymbol{F}'(\{\{v_1\}, \{v_2, v_3\}\})\boldsymbol{A}' = [1\ 0\ 0\ 0\ 0\ 0]\boldsymbol{A}' > 1$, then $\boldsymbol{F}'(\{\{v_1\}, \{v_2\}, \{v_3\}\})\boldsymbol{A}' = [1\ 1\ 0\ 1\ 0\ 0]\boldsymbol{A}' > 1$.)

Solving this integer programming problem, we obtain the optimal value of the objective function $\boldsymbol{H}'\boldsymbol{A}'$, and an optimal solution of $\boldsymbol{A}'$ as follows:

$$\boldsymbol{H}'\boldsymbol{A}' = 0.90, \quad \text{and} \quad \boldsymbol{A}'^T = [0\ 0\ 0\ 1\ 1\ 1].$$

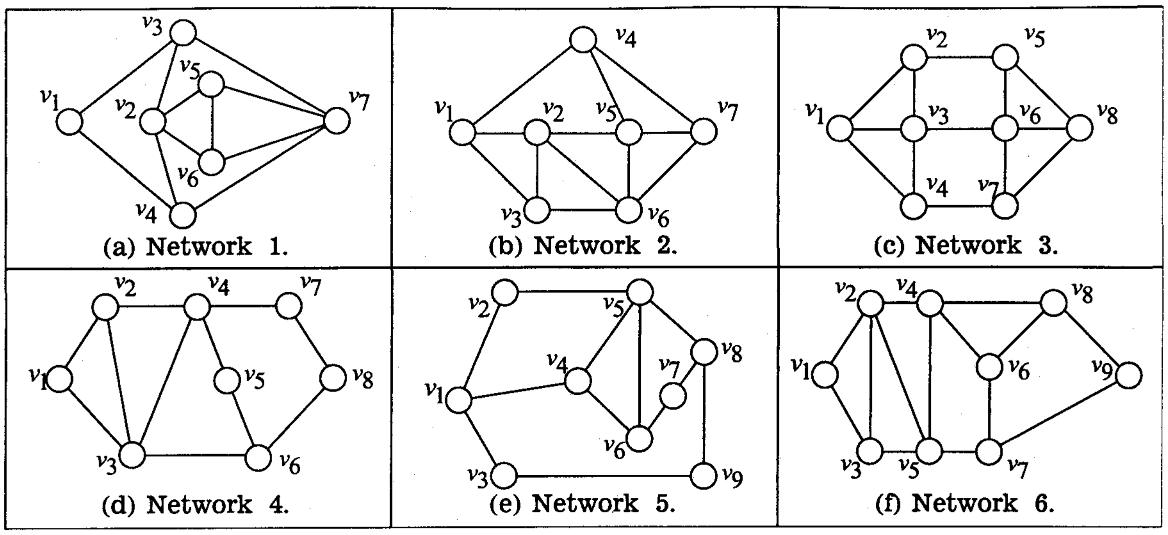

Figure 4.3: Sample networks.

This result means that the optimal availability is equal to 0.90 and that an acceptance set $\mathcal{S}$ that maximizes $\sum_{N \in \mathcal{S}} h(N)$ is $\{\{v_3\}, \{v_1, v_3\}, \{v_1, v_2, v_3\}\}$. Consequently, we obtain the following optimal coterie $\mathcal{C}$ by removing all nonminimal elements in $\mathcal{S}$.

$$\mathcal{C} = \{\{v_3\}\}$$

4.6 Experimental Results

In this section, we present the results of an experiment we conducted for various networks using a SUN Ultra 1 workstation. Figure 4.3 shows the networks used in this experiment, which were taken from [58]. In the experiment, we assumed $p_i = 0.90$ for every node v_i and $p_{i,j} = 0.95$ for every link $e_{i,j}$ just for simplicity.

For each of these networks, we first generated a text file representing the 0-1 integer programming problem by the proposed approach. This process was completed within 10 seconds for any network. The sizes of the formulated problems are shown in Table 4.2. From this table, it is seen that the number of variables and that of inequality constraints are drastically reduced. In the case of Network 5, for example, the number of variables is 205 and that of inequalities is 1389. This means that due to the proposed technique, the variables are reduced by half and the inequality constraints are decreased almost by 97%.

Then, we solved these problems using a linear programming package called *Lindo*.

Table 4.2: Number of variables and number of constraints.

Network	# of variables	# of inequality constraints
1	92	312
2	97	356
3	159	862
4	130	539
5	205	1389
6	229	1670

Lindo needed only a few seconds to solve the problem for any network. The first column of Table 4.3 shows the optimal coteries we obtained. Their availabilities are summarized in the corresponding column of Table 4.4. We also show their *unavailabilities* in Table 4.5. Here we define the *unavailability* of a coterie as 1 - (the availability of the coterie).

Moreover, we evaluated the availabilities (and unavailabilities) of heuristically-designed coteries for comparison by using an evaluation method presented in the next chapter. We chose two heuristic schemes proposed by Barbara and Garcia-Molina[8] and Tong and Kain[64]. Both heuristics are based on voting, and they are designed to construct highly available coteries. Specifically, they explicitly take reliabilities of nodes and links into consideration, unlike other heuristics. In the following, we abbreviate the two heuristics by B&G heuristic and T&K heuristic. We show the results of this evaluation in Tables 4.4 and 4.5.

From these tables, it can be seen that the two heuristics achieve almost the same availability in all cases. In contrast, the difference in the availability (and unavailability) between the optimal coterie and the heuristically-constructed coteries becomes larger when the size of the networks grows. Especially for Network 6, the unavailability of the optimal coterie is only 45 percent of that of B&G coterie. This fact implies that the risk that the mutual exclusion mechanism is not functional can be reduced by half by adopting an optimal coterie. Besides, this improvement of availability is obtained without introducing any additional redundancy. Thus we can conclude that choosing an appropriate coterie is a very significant task for achieving reliable mutual exclusion.

Table 4.3: Optimal coteries.

Network	Optimal Coterie
1	$\{\{v_5, v_6, v_7\}, \{v_3, v_4, v_6, v_7\}, \{v_1, v_4, v_6, v_7\}, \{v_1, v_3, v_6, v_7\}, \{v_2, v_6, v_7\}, \{v_3, v_4, v_5, v_7\}, \{v_1, v_4, v_5, v_7\}, \{v_1, v_3, v_5, v_7\}, \{v_2, v_5, v_7\}, \{v_2, v_4, v_7\}, \{v_2, v_3, v_7\}, \{v_2, v_5, v_6\}, \{v_2, v_3, v_4, v_6\}, \{v_1, v_2, v_4, v_6\}, \{v_1, v_2, v_3, v_6\}, \{v_2, v_3, v_4, v_5\}, \{v_1, v_2, v_4, v_5\}, \{v_1, v_2, v_3, v_5\}\}$
2	$\{\{v_5, v_6, v_7\}, \{v_3, v_4, v_6, v_7\}, \{v_2, v_4, v_6, v_7\}, \{v_1, v_4, v_6, v_7\}, \{v_1, v_3, v_6, v_7\}, \{v_2, v_4, v_5, v_7\}, \{v_2, v_3, v_5, v_7\}, \{v_1, v_2, v_5, v_7\}, \{v_1, v_3, v_4, v_7\}, \{v_1, v_2, v_4, v_7\}, \{v_4, v_5, v_6\}, \{v_1, v_3, v_5, v_6\}, \{v_1, v_3, v_4, v_6\}, \{v_2, v_3, v_6\}, \{v_1, v_2, v_6\}, \{v_2, v_3, v_4, v_5\}, \{v_1, v_3, v_4, v_5\}, \{v_1, v_2, v_4, v_5\}, \{v_1, v_2, v_3, v_5\}\}$
3	$\{\{v_1, v_4, v_6, v_7, v_8\}, \{v_3, v_4, v_5, v_7, v_8\}, \{v_2, v_4, v_5, v_7, v_8\}, \{v_1, v_4, v_5, v_7, v_8\}, \{v_1, v_2, v_5, v_7, v_8\}, \{v_3, v_4, v_6, v_8\}, \{v_2, v_3, v_6, v_8\}, \{v_1, v_3, v_6, v_8\}, \{v_2, v_3, v_5, v_8\}, \{v_2, v_4, v_5, v_6, v_7\}, \{v_1, v_4, v_5, v_6, v_7\}, \{v_3, v_4, v_6, v_7\}, \{v_2, v_3, v_6, v_7\}, \{v_1, v_3, v_6, v_7\}, \{v_3, v_4, v_5, v_6\}, \{v_2, v_3, v_5, v_6\}, \{v_1, v_3, v_5, v_6\}, \{v_1, v_2, v_5, v_6\}, \{v_2, v_3, v_4\}, \{v_1, v_3, v_4\}, \{v_1, v_2, v_4\}, \{v_1, v_2, v_3\}\}$
4	$\{\{v_5, v_6, v_7, v_8\}, \{v_4, v_6, v_7, v_8\}, \{v_2, v_4, v_5, v_7, v_8\}, \{v_1, v_3, v_6, v_8\}, \{v_3, v_4, v_7\}, \{v_1, v_2, v_4, v_7\}, \{v_4, v_5, v_6\}, \{v_1, v_3, v_5, v_6\}, \{v_3, v_4, v_6\}, \{v_2, v_3, v_6\}, \{v_3, v_4, v_5\}, \{v_1, v_2, v_4, v_5\}\}$
5	$\{\{v_2, v_3, v_5, v_7, v_8, v_9\}, \{v_1, v_3, v_4, v_7, v_8, v_9\}, \{v_1, v_2, v_3, v_7, v_8, v_9\}, \{v_1, v_3, v_5, v_8, v_9\}, \{v_4, v_6, v_7, v_8\}, \{v_5, v_6, v_8\}, \{v_4, v_5, v_8\}, \{v_1, v_2, v_5, v_8\}, \{v_5, v_6, v_7\}, \{v_1, v_4, v_6, v_7\}, \{v_1, v_2, v_5, v_6\}, \{v_1, v_3, v_4, v_6\}, \{v_1, v_2, v_4, v_6\}, \{v_2, v_4, v_5\}, \{v_1, v_4, v_5\}\}$
6	$\{\{v_4, v_7, v_8, v_9\}, \{v_6, v_8, v_9\}, \{v_4, v_5, v_7, v_9\}, \{v_1, v_3, v_5, v_7, v_9\}, \{v_2, v_5, v_7, v_9\}, \{v_6, v_7, v_8\}, \{v_4, v_5, v_8\}, \{v_2, v_3, v_4, v_8\}, \{v_1, v_2, v_4, v_8\}, \{v_1, v_3, v_5, v_6, v_7\}, \{v_2, v_5, v_6, v_7\}, \{v_4, v_6, v_7\}, \{v_4, v_5, v_6\}, \{v_2, v_3, v_4, v_6\}, \{v_1, v_2, v_4, v_6\}\}$

Table 4.4: Availabilities of coteries.

Network	Optimal	B&G	T&K
1	0.9838306	0.9819189	0.9819189
2	0.9897070	0.9886633	0.9891055
3	0.9862471	0.9804574	0.9848071
4	0.9770368	0.9617256	0.9683837
5	0.9802050	0.9737180	0.9737180
6	0.9853954	0.9675312	0.9670036

Table 4.5: Unavailabilities of coteries.

Network	Optimal	B&G	T&K
1	1.61694×10^{-2}	1.80811×10^{-2}	1.80811×10^{-2}
2	1.02930×10^{-2}	1.13367×10^{-2}	1.08945×10^{-2}
3	1.37529×10^{-2}	1.95426×10^{-2}	1.51929×10^{-2}
4	2.29632×10^{-2}	3.82744×10^{-2}	3.16163×10^{-2}
5	1.97950×10^{-2}	2.62820×10^{-2}	2.62820×10^{-2}
6	1.46046×10^{-2}	3.24688×10^{-2}	3.29964×10^{-2}

Chapter 5

Evaluation of Dependability Measures

5.1 Introduction

In this chapter, we propose graph-theoretic methods for dependability evaluation of coteries and wr-coteries. As stated before, a number of schemes have been developed for constructing coteries or wr-coteries (e.g., [1, 16, 22, 37, 32, 49, 63]). Coteries generated by these schemes have different advantages and disadvantages of their own. Generally speaking, coteries with high dependability have low performance (e.g. large message-complexity) and vice versa. There are some attempts to evaluate coteries by capturing such a trade-off (e.g., [33, 40, 52]), but these approaches, as well as most of the previous dependability evaluation methods, assume partition-free networks and consider neither the topology of the networks nor link failures (see [5] for a recent survey). Exceptions include [8, 13, 64, 66], where network topologies and link failures are taken into account. In [13] a method based on stochastic Petri nets was proposed for dependability evaluation, while exhaustive state enumeration was used in [8, 64, 66]. However, these approaches are practical only for small systems since they suffer from state explosion.

To overcome these deficiencies of the previous approaches, we newly propose methods for dependability evaluation of coteries and wr-coteries in unreliable networks.

First, we propose a method for evaluating the availability of coteries. As stated before, *availability* is the most common measure for assessing the robustness of quorum-based mutual exclusion mechanisms. This measure is defined as the probability that the critical section can be entered in the presence of failures. We introduce a new graph-theoretic notion called a *Minimal Quorum Spanning Tree* (MQST) and develop the method based on this notion.

Next, for wr-coteries, we propose a method for evaluating *site resiliency*. Site resiliency is the probability that a read or write operation issued by a particular node can be processed successfully in the presence of failures. The proposed site resiliency evaluation method also employs a graph-theoretic approach similar to the proposed availability evaluation method.

The problem of computing the availability or site resiliency is NP-hard[6, 23]. (Notice that computing the probability that there is an operational path between two nodes is already NP-hard[51].) Therefore we show the feasibility of the proposed evaluation methods through running time analysis using moderate-sized networks.

The rest of this chapter is organized as follows: In the next section, we describe the proposed method of evaluating the availability of coteries. In 5.3, we explain the site resiliency evaluation method.

5.2 Availability Evaluation of Coteries

5.2.1 Minimal Quorum Spanning Trees (MQSTs)

The availability of a coterie is equal to the probability that there is a partition group that includes at least one quorum in the coterie. As described in the previous chapter, this is because the nodes in such a partition group can enter the critical section by acquiring permission from a quorum. In graph-theoretical terms, the availability of a coterie $\mathcal{C}$ is the probability that there is a connected subgraph $G' = (V', E')$ of G consisting only of operational nodes and edges, such that $Q \subseteq V'$ for some $Q \in \mathcal{C}$.

Let $SUB(\mathcal{C})$ denote the set of all connected subgraphs of G that include all nodes of at least one quorum in $\mathcal{C}$. It is then clear that the availability of a coterie $\mathcal{C}$ is the probability that at least one subgraph in $SUB(\mathcal{C})$ remains operational. Now we define *Minimal Quorum Spanning Trees* (MQSTs) as follows.

Definition 7 (MQST) An element of $SUB(\mathcal{C})$ is an MQST if and only if no other element of $SUB(\mathcal{C})$ is a proper subgraph of it. (Notice that MQSTs are necessarily trees.)

Example 10 Consider a network in Figure 4.1 and a coterie $\mathcal{C} = \{\{v_3, v_4\}, \{v_2, v_3, v_5\}, \{v_4, v_5\}, \{v_2, v_4, v_6\}, \{v_3, v_5, v_6\}\}$. Suppose that a connected subgraph $(\{v_1, v_2, v_3, v_4\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}\})$ of G remains operational, as in Figure 4.1(b). Since this connected subgraph spans all nodes of a quorum $\{v_3, v_4\}$, the critical section can still be entered in this case. However this is not an MQST because some of its proper connected subgraphs, for example, $(\{v_2, v_3, v_4\}, \{e_{2,3}, e_{2,4}\})$, also include all nodes of this quorum. On the other hand, for $(\{v_2, v_3, v_4\}, \{e_{2,3}, e_{2,4}\})$, none of its proper connected subgraphs spans any quorum in $\mathcal{C}$. Hence this connected subgraph of G is an MQST.

Intuitively speaking, an MQST is a minimal subgraph of G enabling the critical section to be entered. By definition, an MQST is operational if and only if a subgraph in $SUB(\mathcal{C})$ remains operational. Hence the following equation holds.

$$\text{Availability} = \Pr[\text{At least one MQST is operational.}]$$

Based on this equation, we can compute the availability of a coterie by considering only MQSTs, instead of all elements of $SUB(\mathcal{C})$. Specifically, the availability can be calculated in the following two phases:

Phase 1. Enumerate all MQSTs for a given coterie $\mathcal{C}$.

Phase 2. Calculate the probability that at least one MQST is operational.

5.2.2 Phase 1: Enumeration of all MQSTs

For enumeration of all MQSTs, we develop a new algorithm based on breadth-first search. This algorithm, shown in Figure 5.1, generates all MQSTs in nondecreasing order of their sizes, where the size of a tree is the number of edges in it.

First, the algorithm checks whether or not each node in V constitutes a quorum by itself in a given coterie. If the node (say v_a) constitutes a quorum, then $(\{v_a\}, \emptyset)$ is stored in a set $FOUND$ and deleted from TRY. After all elements in TRY have been checked, every node remaining in TRY is expanded by adding each of its adjacent links and nodes. Thus all trees of size 1 such that at least one node in TRY is included are generated.

Then, for each of these trees, whether it spans a quorum is checked. When the nodes of the tree include a quorum, then the tree is stored in $FOUND$ and removed from TRY if none of the previously found trees in $FOUND$ is a subgraph of it; otherwise the tree is simply removed from TRY since it is not minimal. This procedure is repeated for all possible sizes of MQSTs up to $|V| - 1$.

This algorithm thus consists of a *checking* step and an *expanding* step. In the *checking* step, each tree t stored in a set TRY is first tested to determine whether or not it spans all nodes of at least one quorum. Once it is found that t includes a quorum, t is deleted from TRY and t will not be expanded further because any tree containing t as a proper subgraph is not an MQST. Then, whether t is an MQST is determined by checking whether none of the previously found MQSTs, stored in a set $FOUND$, is a subgraph of t. If a subgraph of t already exists in $FOUND$, then t is not an MQST. Otherwise t is an MQST, thus it is added to $FOUND$.

Once the checking process has been completed, only trees spanning none of the quorums remain in the set TRY. Then the *expanding* process is performed in order to increase the size of each tree in TRY by adding a new adjacent vertex to it. After increasing the size of each tree in TRY by one, the checking process is performed again. When no further expansion is possible, the algorithm terminates.

5.2.3 Phase 2: Computation of the Availability

Once all MQSTs have been found, the next phase is to calculate the probability that at least one of them is operational.

Let u_j and $u_{j,k}$ be Boolean variables that represent the event that a node v_j is operational and the event that a link $e_{j,k}$ is operational, respectively. Then, the event that all components of an MQST (V', E') are operational is represented by a Boolean product

$$\bigwedge_{v_j \in V'} u_j \wedge \bigwedge_{e_{j,k} \in E'} u_{j,k}.$$

Hence, the event that for at least one MQST, its all components are operational is represented by the following Boolean sum of products.

$$\bigvee_{\text{for every MQST}(V',E')} \left(\bigwedge_{v_j \in V'} u_j \wedge \bigwedge_{e_{j,k} \in E'} u_{j,k} \right)$$

```
/* Step 1: Initialization */
  $TRY := \{(\{v_1\}, \emptyset), \cdots, (\{v_n\}, \emptyset)\}$
  $FOUND := \emptyset$
/* Step 2: Generation of all MQSTs */
  While ($TRY \neq \emptyset$) do
    /* 2.1 Checking Step */
      For all $t \in TRY$ do
        If ($t$ spans all nodes in at least one quorum of the given coterie $\mathcal{C}$) Then
          If (any tree in $FOUND$ is not a subgraph of $t$)
            Then
              add $t$ to $FOUND$
              remove $t$ from $TRY$
            Else
              remove $t$ from $TRY$
          END_If
        END_If
      END_For
    /* 2.2 Expanding Step */
      $NEW := \emptyset$
      For all $t = (V', E') \in TRY$ do
        $AE$ := set of all adjacent edges to $t$
        $AE' := \{e_{a,b} \in AE | (v_a \in V' \wedge v_b \notin V') \vee (v_b \in V' \wedge v_a \notin V')\}$
        For all ($e_{a,b} \in AE'$) do
          $newt := (V' \cup \{v_a, v_b\}, E' \cup \{e_{a,b}\})$
          add $newt$ to $NEW$
        END_For
      END_For
      $TRY := NEW$
  END_While
```

Figure 5.1: Algorithm for MQST enumeration.

Obviously, the availability is equal to the probability that the event represented by this Boolean sum of products occurs. However, computing this probability cannot be done in a straightforward fashion because these products usually share common variables. In order to compute the probability, therefore, we first transform the Boolean expression into another equivalent Boolean sum of mutually exclusive products. Then, we can obtain this probability as the sum of the probabilities of the events corresponding to these exclusive products.

Fortunately, techniques for such a transformation have already been studied well, and several algorithms, such as SYREL[25] and CAREL[58], have been proposed. These algorithm, called terminal reliability algorithms, were originally developed for computing the probability that there exists an operational path between two nodes in an unreliable network. In our experiments, we implemented the algorithm SYREL[25] and used it in Phase 2. For example, if the MQSTs are $(\{v_1, v_2\}, \{e_{1,2}\})$, $(\{v_1, v_3\}, \{e_{1,3}\})$, and $(\{v_2, v_3\}, \{e_{2,3}\})$, then the event that at least one MQST is operational is represented by a Boolean sum of products $u_1u_2u_{1,2} \vee u_1u_3u_{1,3} \vee u_2u_3u_{2,3}$. SYREL transforms this expression into an equivalent Boolean sum of mutually exclusive products $u_1u_2u_{1,2} \vee u_1u_3u_{1,3}(\overline{u_2u_{1,2}}) \vee u_2u_3u_{2,3}(\overline{u}_1) \vee u_2u_3u_{2,3}(u_1\overline{u}_{1,2}\overline{u}_{1,3})$.* From this expression, the availability is directly obtained as follows: $p_1p_2p_{1,2} + p_1p_3p_{1,3}(1 - p_2p_{1,2}) + p_2p_3p_{2,3}(1 - p_1) + p_2p_3p_{2,3}(p_1(1 - p_{1,2})(1 - p_{1,3}))$.

Different terminal reliability algorithms have different time complexity. For example, the complexity of SYREL is $O(m^2)$ [25], where m denotes the number of MQSTs.

Example 11 Consider again the network shown in Figure 4.1 and coterie $\mathcal{C} = \{\{v_3, v_4\}, \{v_2, v_3, v_5\}, \{v_4, v_5\}, \{v_2, v_4, v_6\}, \{v_3, v_5, v_6\}\}$. Applying the algorithm shown in Figure 5.1, we find that there exist a total of nine MQSTs, shown in Figure 5.2. Using SYREL and assuming $p_i = 0.9$ and $p_{i,j} = 0.9$ for every node v_i and link $v_{i,j}$, the probability that at least one of the MQSTs is operational is found to be 0.9646616. This probability is equal to the availability of the coterie $\mathcal{C}$.

*The details of the algorithm are omitted here. Interested readers are referred to [25].

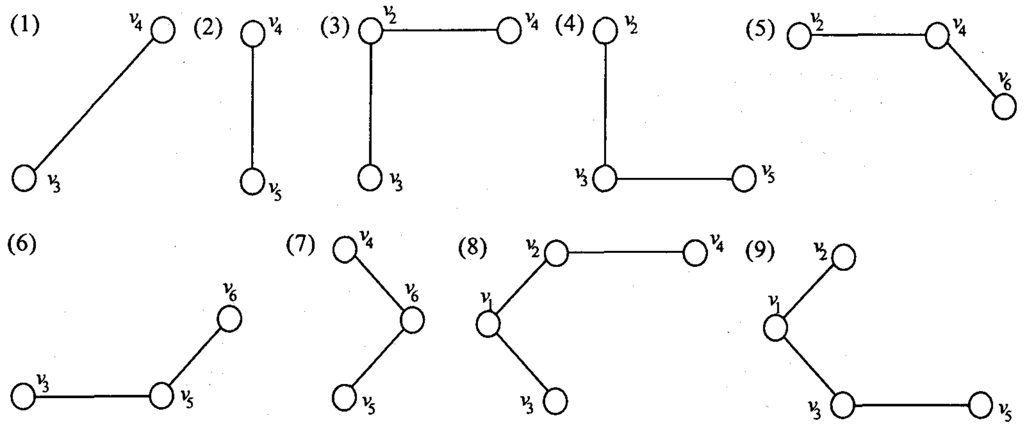

Figure 5.2: MQSTs.

5.2.4 Running Time Evaluation

In this section, we show the results of running time evaluation of the proposed evaluation method. We demonstrate the feasibility of the method via this evaluation, since computing the availability of coteries in unreliable networks is NP-hard[23]. We implemented the proposed method in C language and conducted the evaluation in a UNIX environment on a Sun Ultra 1 workstation.

We used a collection of networks from [14] as sample networks. For each network, we evaluated the availability of a coterie constructed by majority voting[63]. This coterie is composed of all node groups consisting of exactly $\lceil\frac{(n+1)}{2}\rceil$ nodes. For example, when $V = \{v_1, v_2, v_3\}$, this coterie is $\{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}\}$. There are several reasons for choosing this coterie. First, it has been well-studied in the literature (e.g., [8, 23, 63]). Second, the coterie is defined uniquely regardless of the topology of the network. In addition, this coterie has the largest number of quorums among all possible coteries, if the number of nodes is odd. Roughly speaking, therefore, the evaluation of this coterie is relatively difficult, compared to other coteries.

Table 5.1 show the result of the evaluation. In this table, the time needed for Phase 1 and Phase 2, the total running time, the number of MQSTs, and the availability of the coterie are shown for each network. The availability was calculated assuming that all nodes and links have the same reliability 0.9. (Needless to say, this assumption was made for simplicity only. The actual reliabilities of components do not have any effect on the

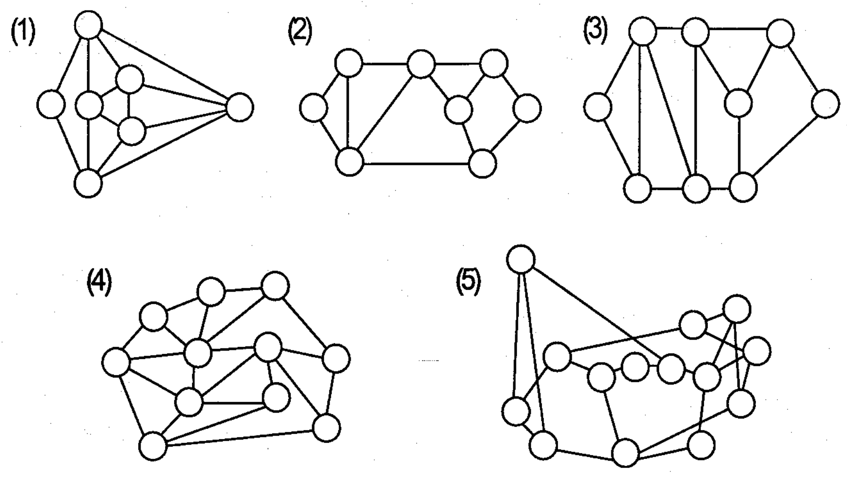

Figure 5.3: Sample networks.

running time of the proposed method.)

The table shows that the running time increases rapidly as the size of the network grows. Obviously, this is because the number of MQSTs increases exponentially in the network size. For all the networks, however, the proposed method completed the evaluation within a sufficiently admissible amount of time.

It is also seen that the time needed for the second phase was dominant in the total running time. Though we used SYREL[25] for this phase in this experiment, other terminal algorithms proposed more recently, such as CAREL[58], might run faster. So we conjecture that there is still room for further improvement of the running time.

For comparison, we also implemented a program that evaluates the availability by exhaustive enumeration of network states (i.e., configurations). The pseudo-code of this program is shown in Figure 5.4. We also slightly modified the algorithm for computing $h(N)$ shown in Figure 4.2, and conducted the evaluation using this algorithm. (Notice that the availability is $\sum_{\{N \mid \exists Q \in \mathcal{C}, Q \subseteq N \subseteq V\}} h(N)$.) Table 5.2 shows the running times of the two enumerative methods. When the number of nodes is 10, the state enumeration-based method could not complete the evaluation within 24 hours. These results clearly show that the proposed method based on MQSTs is superior to the other methods.

Table 5.1: Running times of the proposed method.

Network	Phase 1	Phase 2	Total Time	# of MQSTs	Availability
1	0.02s	0.07s	0.09s	111	0.99013
2	0.03s	0.05s	0.08s	126	0.94505
3	0.05s	0.13s	0.18s	182	0.95621
4	5.31s	142.98s	148.29s	2595	0.99078
5	20.59s	701.61s	722.20s	4265	0.97665

Table 5.2: Running times of state enumeration-based method and the subgraph enumeration method.

Network	State enumeration	Subgraph enumeration
1	11.36s	0.51s
2	12.91s	0.48s
3	124.63s	1.73s
4	89331.8s	231.3s
5	N/A	851.81s

```
$Availability := 0$
For (every state $(V'(\subseteq V), E'(\subseteq E))$ do
  /* Determination of Partition Groups in $(V', E')$ */
    $N := V', Set\_PGs := \emptyset$
    While ($N$ is not empty) do
      Select one node $v_i$ from $N$
      $PG := \{v_i\}$, $N := N - \{v_i\}$
      $AN := \{v_j \in N | v_j$ is adjacent to $v_i\}$
      While ($AN$ is not empty) do
        $PG := PG + AN$, $N := N - AN$
        $AN := \{v_j \in N | v_j$ is adjacent to a node in $PG\}$
      END_While
      $Set\_PGs := Set\_PGs + PG$
    END_While
  /* Accumulation of availability */
    If (there are $g \in Set\_PGs$ and $q \in C$ such that $q \subseteq g$) do
      $prob := \prod_{v_i \in V'} p_i \prod_{v_i \in V-V'} (1 - p_i) \prod_{e_{i,j} \in E'} p_{i,j} \prod_{e_{i,j} \in E-E'} (1 - p_{i,j})$
      $Availability := Availability + prob$
    END_If
END_For
```

Figure 5.4: Evaluation algorithm by state enumeration.

5.3 Site Resiliency Evaluation

In this section, we address the issue of dependability evaluation of wr-coteries. In order to assess the dependability of replication control schemes based on wr-coteries, several measures have been proposed so far[7, 40, 49]. Here we select *site resiliency*[49] as a measure to be evaluated because its concept is natural and it clearly takes into consideration the difference between read operations and write operations. Site resiliency is the probability that a node that wishes to perform a read or write operation can successfully find a group of operational nodes that enables the operation by agreeing to it. In other words, this measure represents the availability of each operation. Based on graph theory, we first show that such a group of operational nodes can be found if and only if a tree satisfying some conditions remains operational on the network graph. Then, we propose a new algorithm for enumerating all trees that satisfy these conditions. Once the trees are found, the site resiliency can be obtained by computing the probability that at least one of these trees is operational.

In the rest of this section, we assume that a fraction r, $0 \leq r \leq 1$, of all operations

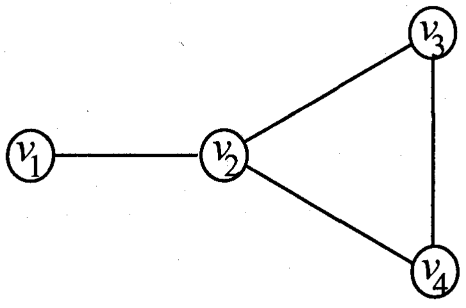

Figure 5.5: Network with four nodes.

are read operations, and a fraction $w(=1-r)$ of all operations are write operations.

5.3.1 Site Resiliency

The site resiliency of a particular node v_i is the probability that if node v_i initiates a read or write operation, then it can acquire permission for the operation given that v_i is operational[49]. Formally, it is defined as follows:

Definition 8 (Site Resiliency) Suppose that a wr-coterie $(\mathcal{W}, \mathcal{R})$ is adopted. Let $P_i(\mathcal{D})$ $(\mathcal{D} \in \{\mathcal{W}, \mathcal{R}\})$ be the conditional probability that node v_i can acquire permission from all nodes in at least one quorum in $\mathcal{D}$, given that v_i is operational. The site resiliency of v_i, $Resil_i$, is

$$Resil_i = rP_i(\mathcal{R}) + wP_i(\mathcal{W}).$$

Example 12 Consider a small network shown in Figure 5.5 and assume $r = w = 0.5$, $p_i = 0.9$ for every $v_i \in V$, and $p_{i,j} = 0.9$ for every $e_{i,j} \in E$. When a wr-coterie $(\mathcal{W} = \{\{v_1, v_2, v_4\}, \{v_2, v_3, v_4\}\}, \mathcal{R} = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_4\}\})$ is adopted, the site resiliency of v_1 is determined as follows: From Figure 5.5, one can see that node v_1 can get permission from a read quorum in $\mathcal{R}$ if and only if nodes v_1 and v_2 and link $e_{1,2}$ are operational. When node v_1 is known to be operational, the probability that all of these components are operational is $p_2p_{1,2} = 0.81(= P_1(\mathcal{R}))$. Similarly, node v_1 can get permission from a write quorum in $\mathcal{W}$ if and only if there is an operational path from v_1 to v_4. The conditional probability that such an operational path exists when v_1 is known to be operational is

$p_{1,2}p_2(p_{2,4} + (1 - p_{2,4})p_{2,3}p_3p_{3,4})p_4 = 0.7092 (= P_1(\mathcal{W}))$. Then, the site resiliency of v_1 is $Resil_1 = rP_1(\mathcal{R}) + wP_1(\mathcal{W}) = 0.7596$.

Clearly, calculation in such an ad hoc manner is possible only for very small systems like the one in the above example. In the next section, we explain the proposed method that computes site resiliency in a systematic fashion.

5.3.2 Basic Analysis

In order to evaluate site resiliency analytically, we first investigate graph-theoretical properties that necessarily hold when an operation can be processed successfully. In this section, we will not distinguish the read quorum set from the write quorum set, and represent the quorum set of interest by $\mathcal{D}$ without loss of generality.

As described in the previous chapter, we assume that a node v_i can acquire permission from a node v_j if and only if there is an operational path from v_i to v_j. This means that if there is an operational connected subgraph containing node v_i, then v_i can attain permission from all nodes in the subgraph.

Now let $SUB_i(\mathcal{D})$ be the set of all connected subgraphs of G that include v_i and all nodes of at least one quorum in $\mathcal{D}$. It is then clear that v_i can acquire permission from all nodes of at least one quorum if and only if at least one subgraph in $SUB_i(\mathcal{D})$ remains operational.

Next, we consider the minimality of subgraphs in $SUB_i(\mathcal{D})$. We say that a subgraph $K \in SUB_i(\mathcal{D})$ of G is minimal if there is no element $K' \in SUB_i(\mathcal{D})$ such that K' is a proper subgraph of K. Let $\mathcal{M}_i(\mathcal{D})$ be the set of all minimal subgraphs in $SUB_i(\mathcal{D})$. Then it is seen that v_i can acquire permission from all nodes of at least one quorum if and only if at least one subgraph in $\mathcal{M}_i(\mathcal{D})$ remains operational.

By definition, any subgraph T in $\mathcal{M}_i(\mathcal{D})$ is a tree. Hence, $\mathcal{M}_i(\mathcal{D})$ is the set of *minimal* trees such that v_i and at least one quorum in $\mathcal{D}$ are included.

Example 13 Consider the network shown in Figure 5.5 again and suppose that the wr-coterie $(\mathcal{W} = \{\{v_1, v_2, v_4\}, \{v_2, v_3, v_4\}\}, \mathcal{R} = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_4\}\})$ is adopted. Then, all elements in $\mathcal{M}_1(\mathcal{R})$ are

$$(\{v_1, v_2\}, \{e_{1,2}\}),$$

while all elements in $\mathcal{M}_1(\mathcal{W})$ are

$$(\{v_1, v_2, v_4\}, \{e_{1,2}, e_{2,4}\})$$
$$(\{v_1, v_2, v_3, v_4\}, \{e_{1,2}, e_{2,3}, e_{3,4}\}).$$

It is then clear that the following equation holds.

$$P_i(\mathcal{D}) = \Pr[\text{At least one tree in } \mathcal{M}_i(\mathcal{D}) \text{ is operational. } | \ v_i \text{ is operational.}]$$

Due to this equation, $P_i(\mathcal{D})$, that is, the probability that node v_i can acquire permission from at least one quorum in $\mathcal{D}$, can be computed in the following two phases:

Phase 1. Enumerate all trees in $\mathcal{M}_i(\mathcal{D})$.

Phase 2. Calculate the conditional probability $P_i(\mathcal{D})$.

Once $P_i(\mathcal{R})$ and $P_i(\mathcal{W})$ are calculated, the site resiliency of v_i can easily be obtained, since it is equal to $rP_i(\mathcal{R}) + wP_i(\mathcal{W})$. (Note that when the read and write quorum sets are the same, computation of $P_i(\mathcal{W})$ can be omitted.) In the rest of this section, we explain the proposed method for calculating $P_i(\mathcal{D})$.

5.3.3 Phase 1: Enumeration of trees in $\mathcal{M}_i(\mathcal{D})$

For enumeration of all trees in $\mathcal{M}_i(\mathcal{D})$, we develop an algorithm based on breadth-first search, similar to the algorithm shown in 5.1. Figure 5.6 shows the pseudo-code of the algorithm. This algorithm generates all trees in $\mathcal{M}_i(\mathcal{D})$ in nondecreasing order of their size, where the size of a tree is defined as the number of edges in it.

First, the algorithm checks whether node v_i is a quorum in $\mathcal{D}$. If so, then $(\{v_i\}, \emptyset)$ is the only element in $\mathcal{M}_i(\mathcal{D})$. Otherwise, v_i is expanded by adding each of its adjacent links and nodes. Thus trees of size 1 where node v_i is included are generated. Then, for each of these trees, whether or not its nodes include a quorum is checked. This procedure is repeated for all possible sizes of trees in $\mathcal{M}_i(\mathcal{D})$ up to $|V| - 1$.

```
/* Step 1: Initialization */
  $TRY := \{(\{v_i\}, \emptyset)\}$
  $FOUND := \emptyset$
/* Step 2: Generation of all trees in $\mathcal{M}_i(\mathcal{D})$ */
  While ($TRY \neq \emptyset$) do
    /* 2.1 Checking Step */
      For all $t \in TRY$ do
        If ($t$ includes at least one quorum in $\mathcal{D}$) Then
          If (any tree in $FOUND$ is not a subgraph of $t$)
            Then
              add $t$ to $FOUND$
              remove $t$ from $TRY$
            Else
              remove $t$ from $TRY$
          END_If
        END_If
      END_For
    /* 2.2 Expanding Step */
      $NEW := \emptyset$
      For all $t = (V', E') \in TRY$ do
        $AE$ := set of all adjacent edges to $t$
        $AE' := \{e_{a,b} \in AE | (v_a \in V' \wedge v_b \notin V') \vee (v_b \in V' \wedge v_a \notin V')\}$
        For all ($e_{a,b} \in AE'$) do
          $newt := (V' \cup \{v_b\}, E' \cup \{e_{a,b}\})$
          add $newt$ to $NEW$
        END_For
      END_For
      $TRY := NEW$
  END_While
```

Figure 5.6: Algorithm for enumeration of trees in $\mathcal{M}_i(\mathcal{D})$.

5.3.4 Phase 2: Calculation of $P_i(\mathcal{D})$

Once all trees in $\mathcal{M}_i(\mathcal{D})$ have been found, the next phase is to calculate the conditional probability that at least one of them is operational given that node v_i is operational.

Let u_j and $u_{j,k}$ be Boolean variables that represent the event that a node v_j is operational and the event that a link $e_{j,k}$ is operational, respectively. Then, the event that all components of a tree (V', E') except for v_i are operational is represented by a Boolean product

$$\bigwedge_{v_j \in V' - \{v_i\}} u_j \wedge \bigwedge_{e_{j,k} \in E'} u_{j,k}.$$

Hence, the event that for at least one tree in $\mathcal{M}_i(\mathcal{D})$, all of its components except v_i are operational is represented by the following Boolean sum of products.

$$\bigvee_{(V',E') \in \mathcal{M}_i(\mathcal{D})} (\bigwedge_{v_j \in V' - \{v_i\}} u_j \wedge \bigwedge_{e_{j,k} \in E'} u_{j,k}).$$

Obviously, $P_i(\mathcal{D})$ is equal to the probability that the event represented by this Boolean sum of products occurs. As described before, we can compute this probability by applying a terminal reliability algorithm, such as SYREL[25] or CAREL[58].

5.3.5 Case Study: Determining Optimal Thresholds of the Weighted Voting Scheme

In this section, we demonstrate the use of the evaluation method for optimizing the site resiliency of a well-known replication control scheme called the weighted voting scheme[22].

Weighted Voting

In the weighted voting scheme, each node is assigned zero or more votes. Let $x_i (\geq 0)$ denote the number of votes assigned to node v_i. Any node that wants to perform a read operation on the data must first collect at least a predefined number of votes from the nodes in the system before it can actually read the data. Let R denote the predefined number. Similarly, a node must first acquire at least another predefined number of votes before it can write the data. Let W denote the predefined number. R and W are called the *read threshold* and the *write threshold*, respectively. Let X be the total number of votes of all nodes, i.e.,

$$X = \sum_{v_i \in V} x_i.$$

In replicated database systems, simultaneous read and write operations, and simultaneous write and write operations on replicas must be disallowed to preserve their consistency. To achieve this, these thresholds must satisfy the following two conditions:

(1) $R + W > X$, and

(2) $2W > X$.

The first condition guarantees that a read operation and a write operation cannot be performed concurrently. Similarly, the second condition guarantees that two write operations cannot be performed concurrently.

The assignment of votes and thresholds determines a wr-coterie implicitly. That is, if a node group has R (or W) votes or more, the node group contains a read (or write) quorum.

Example 14 Consider the network shown in Figure 5.5 and suppose that each of v_1, v_2, and v_3 has one vote, while v_4 is assigned two votes. Let the read threshold R be 2 and the write threshold W be 4. These satisfy the conditions on a threshold assignment. Then this vote and threshold assignment defines the following wr-coterie.

$$\begin{aligned}(\mathcal{W}, \mathcal{R}) &= \{\{v_1, v_2, v_4\}, \{v_1, v_3, v_4\}, \{v_2, v_3, v_4\}\} \\ &\quad \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_4\}\}\end{aligned}$$

Determination of Optimal Thresholds

By applying the site resiliency evaluation method to all possible pairs of the threshold values, we can determine optimal thresholds that maximize dependability. In this section, we demonstrate such an application of the evaluation method to dependability optimization.

Formally, the optimization problem we discuss here is stated as follows:

> Given a network graph $G = (V, E)$, the probability of each component being operational $\{p_i | v_i \in V\}$ and $\{p_{i,j} | e_{i,j} \in E\}$, the number of votes assigned to each node $\{x_i | v_i \in V\}$, and the probability of read operation r, determine

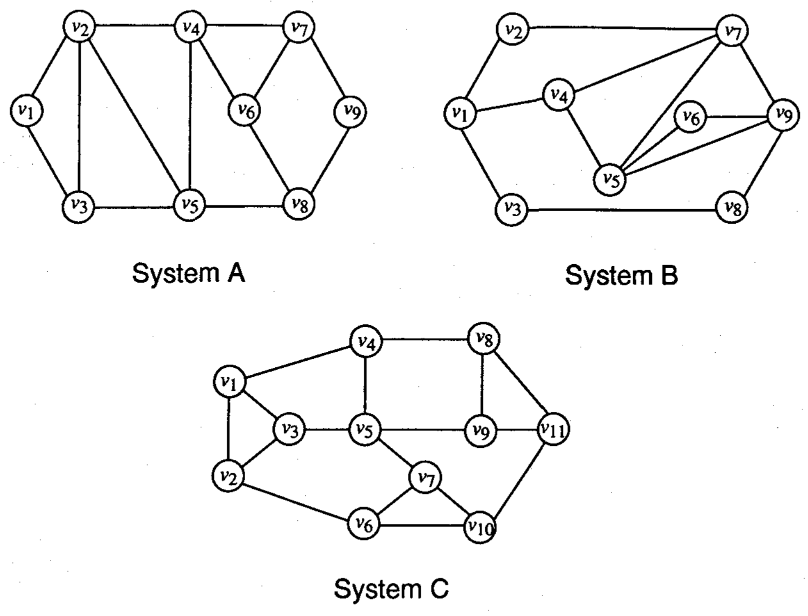

Figure 5.7: Examples of networks.

the values of the read threshold R and the write threshold W such that an objective function is maximized.

As the objective function, we consider the average value of the site resiliency of all nodes in a system. Let *Av_Resil* denote this measure. Formally *Av_Resil* can be written as follows:

$$Av\_Resil = \frac{1}{|V|} \sum_{v_i \in V} Resil_i.$$

As examples, we consider three networks shown in Figure 5.7. (The topologies of these networks were taken from [47] and [58].) We assume that for every $v_i \in V$, $p_i = 0.99$ and that for every $e_{i,j} \in E$, $p_{i,j} = 0.97$. These values were used for modeling a real LAN[24]. To determine a vote assignment, we used a heuristic method proposed in [64]. In this case, since for any node the probability of being operational is the same and high, this heuristic assigns one vote to every node for all the networks. That is, we assume $x_i = 1$ for any v_i.

To find optimal values of the two thresholds, we implemented the proposed method using C language and computed *Av_Resil* for all possible pairs of the two values, i.e., R and W. For example, since in System A the total number of votes is equal to nine, all possible pairs of the thresholds in this system are as follows:

$$(1,9),(2,8),(3,7),(4,6),(5,5).$$

Table 5.3 shows the results of this computation. (The running times are shown in Tables 5.6, 5.7, and 5.8.) Similarly, Tables 5.4 and 5.5 show the results for System B and System C, respectively. In the tables, figures written in boldface indicate optimal values of *Av_Resil* for various values of r.

Form these tables, optimal values of the thresholds can easily be found for each value of r. In System A, for example, when $r = 0.99$, the optimal values of R and W are two and eight, respectively. In contrast, when r is equal to or less than 0.60, *Av_Resil* is maximized by assigning the same value to both R and W. For Systems B and C, the optimal values of the thresholds vary similarly, according to the probability of read operations, r.

Interestingly, even when r is 0.99, a well-known scheme called the read one write all (ROWA) scheme[26] is not effective for all the systems. (In this case, the weighted voting scheme is equivalent to ROWA when R is one and W is equal to the total number of votes.) This is because in ROWA any node failure or network partition blocks write operations, and the probability that all nodes are connected via reliable links becomes rather low.

It is also seen that the values of *Av_Resil* in the case of System C are considerably higher than System A and System B. This is due to the fact that System C has larger degree of replication. More precisely, this can be explained as follows: When the number of replicas increases, the number of trees that enable read or writes operations also increases. As a result, the site resiliency improves in spite of an increase in the number of necessary votes. However, when a system has a smaller number of edges, it may not exhibit such scalability. For example, imagine a system modeled by a single path, and let v_1 be the node located at one end of the system. In this case, $\mathcal{M}_1(\mathcal{R})$ and $\mathcal{M}_1(\mathcal{W})$ have only one element regardless of the values of the thresholds. This means that even if the size of the system, that is, the degree of replication increases, the site resiliency never improves. Though this is an extreme case, a similar situation occurs when the topology of the system is a ring or a tree.

Table 5.3: Average values of site resiliency for System A.

(R, W)	$r = 0.99$	$r = 0.95$	$r = 0.90$	$r = 0.80$	$r = 0.70$	$r = 0.60$	$r = 0.50$	$r = 0.30$
(1, 9)	0.99921	0.99604	0.99208	0.98417	0.97625	0.96833	0.96042	0.94458
(2, 8)	**0.99957**	**0.99939**	0.99915	0.99868	0.99821	0.99774	0.99727	0.99634
(3, 7)	0.99948	0.99942	0.99936	0.99922	0.99908	0.99895	0.99881	0.99854
(4, 6)	0.99928	0.99926	**0.99923**	**0.99916**	**0.99909**	0.99902	0.99895	0.99881
(5, 5)	0.99890	0.99890	0.99890	0.99890	0.99890	**0.99890**	**0.99890**	**0.99890**

Table 5.4: Average values of site resiliency for System B.

(R, W)	$r = 0.99$	$r = 0.95$	$r = 0.90$	$r = 0.80$	$r = 0.70$	$r = 0.60$	$r = 0.50$	$r = 0.30$
(1, 9)	0.99918	0.99592	0.99185	0.98370	0.97554	0.96739	0.95924	0.94293
(2, 8)	**0.99922**	0.99898	0.99867	0.99806	0.99744	0.99683	0.99621	0.99498
(3, 7)	0.99890	**0.99886**	**0.99880**	**0.99868**	0.99857	0.99846	0.99834	0.99812
(4, 6)	0.99885	0.99884	0.99883	0.99880	**0.99878**	0.99875	0.99873	0.99868
(5, 5)	0.99877	0.99877	0.99877	0.99877	0.99877	**0.99877**	**0.99877**	**0.99877**

Table 5.5: Average values of site resiliency for System C.

(R, W)	$r = 0.99$	$r = 0.95$	$r = 0.90$	$r = 0.80$	$r = 0.70$	$r = 0.60$	$r = 0.50$	$r = 0.30$
(1, 11)	0.99904	0.99520	0.99040	0.98081	0.97121	0.96162	0.95202	0.93283
(2, 10)	0.99990	0.99971	0.99947	0.99900	0.99853	0.99805	0.99758	0.99663
(3, 9)	**0.99991**	**0.99989**	**0.99986**	0.99982	0.99977	0.99972	0.99967	0.99957
(4, 8)	0.99986	0.99985	0.99984	0.99983	0.99981	0.99979	0.99978	0.99974
(5, 7)	0.99984	0.99984	0.99984	**0.99983**	0.99983	0.99982	0.99982	0.99981
(6, 6)	0.99983	0.99983	0.99983	0.99983	**0.99983**	**0.99983**	**0.99983**	**0.99983**

Table 5.6: Running time per node in System A.

(R, W)	for $P_i(\mathcal{R})$	for $P_i(\mathcal{W})$	for $Resil_i$
(1, 9)	0.04	2.84	2.88
(2, 8)	0.04	4.11	4.15
(3, 7)	0.04	1.53	1.57
(4, 6)	0.06	0.33	0.39
(5, 5)	0.10	—	0.10

(Unit: Second)

Table 5.7: Running time per node in System B.

(R, W)	for $P_i(\mathcal{R})$	for $P_i(\mathcal{W})$	for $Resil_i$
(1, 9)	0.03	0.62	0.65
(2, 8)	0.04	0.91	0.95
(3, 7)	0.04	0.47	0.51
(4, 6)	0.05	0.16	0.21
(5, 5)	0.07	—	0.07

(Unit: Second)

Table 5.8: Running time per node in System C.

(R, W)	for $P_i(\mathcal{R})$	for $P_i(\mathcal{W})$	for $Resil_i$
(1, 11)	0.04	492.03	492.07
(2, 10)	0.04	805.82	805.86
(3, 9)	0.04	203.69	203.73
(4, 8)	0.05	29.44	29.49
(5, 7)	0.10	3.04	3.14
(6, 6)	0.38	—	0.38

(Unit: Second)

Chapter 6

Extension of Failure Model

6.1 Introduction

In Chapters 4 and 5, we assumed that each component was either operational or failed. However, this assumption may result in an underestimate of the dependability because communication may be possible via a computing node even when this node is not fully operational. This is the case when, for example, the software process in the node fails but the computing node is still working. In such a situation, though the node can no longer request or grant permission, it can still relay messages issued by other nodes. Thus, even when intermediate nodes are not fully operational, if they can still correctly relay messages, two distinct operational nodes can communicate mutually via these nodes.

To take such a situation into consideration, we introduce a new failure model in this chapter. In the new failure model, we consider a degraded operational mode of nodes, in addition to an operational mode and a failed mode. We assume that when a node is in this degraded operational mode, the node neither requests to enter the critical section nor gives permission to other nodes, but communication via the node is possible.

6.2 An Extended Failure Model

We assume that a node is in one of the three modes given below.

- Fully operational mode.

- Degraded operational mode: If a node is in this mode, the node neither requests to enter the critical section nor grants permission to other nodes, but communication via this node is possible.

- Completely failed mode.

On the other hand, for links, we assume a conventional two-mode failure model, that is, we assume that each link is either operational or failed. Failures of nodes and links are assumed to be mutually independent.

For each node v_i, the probability of being in the fully operational mode is denoted by p_i. Similarly, that of being in the degraded operational mode is denoted by q_i. For each link $v_{i,j}$, the probability of being operational is represented by $p_{i,j}$. The failure model based on the assumption of two-mode failures is a special case of the three-mode failure model in which $q_i = 0$ for every node v_i.

Under the new failure model, we can represent the status of the network G as a triplet (VO, VD, EO) where VO is the set of all fully-operational nodes, VD is the set of all degraded-operational nodes, and EO is the set of all operational links.

Example 15 Consider a network $G = (V, E)$ shown in Figure 4.1(a). Figure 6.1 schematically represents the status of the network where v_3 is in the completely failed mode, v_5 is in the degraded operational mode, and links $e_{2,4}$ and $e_{4,6}$ are completely failed.

In the previous chapter, we defined a partition group as a maximal node group such that all nodes in that group can communicate with each other. We can adopt this notion

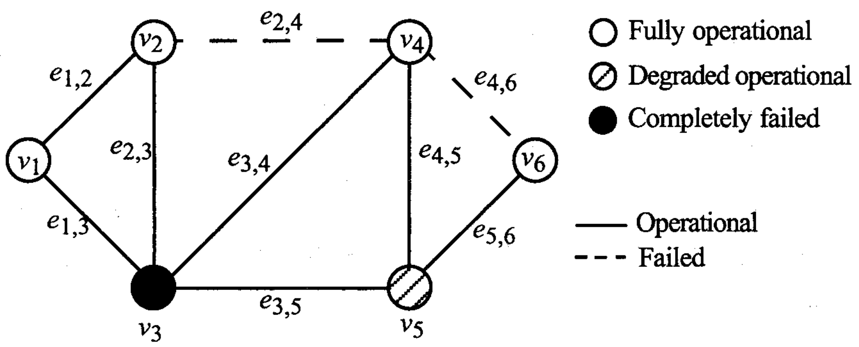

Figure 6.1: Three-mode failures of nodes.

in the same way under the new failure model. For instance, in Example 15, $\{v_1, v_2\}$ and $\{v_4, v_6\}$ are the only partition groups.

6.3 Augmenting the Network Graph

It is not straightforward to adapt the techniques proposed in the previous chapters to the new failure model because they are based on the assumption that each node or link has only two states. To do so, we introduce another undirected graph $AG = (AV, AE)$.

$AG = (AV, AE)$ is an augmented graph of $G = (V, E)$, and it is generated by adding a new vertex v_{n+i} and a new edge $e_{i,n+i}$ to each node $v_i \in V$, i.e., $AV = V \cup \{v_{n+1}, v_{n+2}, \cdots, v_{2n}\}$ and $AE = E \cup \{e_{1,n+1}, e_{2,n+2}, \cdots, e_{n,2n}\}$. In order to represent the status of the network by AG, we assign one of the two states *up* and *down* to every vertex and edge according to the following rules. Here f_i and $f_{i,j}$ denote the state of v_i and that of $e_{i,j}$.

$$\text{For } v_i \in V, \qquad f_i = \begin{cases} up & v_i \in VO \cup VD \\ down & \text{otherwise} \end{cases}$$

$$\text{For } v_i \in AV - V, \quad f_i = \begin{cases} up & v_{i-n} \in VO \\ down & \text{otherwise} \end{cases}$$

$$\text{For } e_{i,j} \in E, \qquad f_{i,j} = \begin{cases} up & e_{i,j} \in EO \\ down & \text{otherwise} \end{cases}$$

$$\text{For } e_{i,j} \in AE - E, \quad f_{i,j} = \begin{cases} up & v_i \in VO \\ down & \text{otherwise} \end{cases}$$

Intuitively, in the augmented graph AG, vertex v_i with $i \leq n$ represents message relay function of node v_i, while vertex v_{n+i} represents the function for requesting the critical section and granting permission to other nodes. The state of the node v_i is thus represented by a small subgraph $(\{v_i, v_{n+i}\}, \{e_{i,n+i}\})$ of AG. (Edge $e_{i,n+i}$ has no physical meaning.)

For $AV' \subseteq AV$ and $AE' \subseteq AE$, we call a tuple (AV', AE') a *configuration* of AG. The *current configuration* is defined as a configuration (AV', AE') such that AV' and AE' are the set of all currently up vertices and the set of all currently up edges. Thus the current configuration represents the current status of the network.

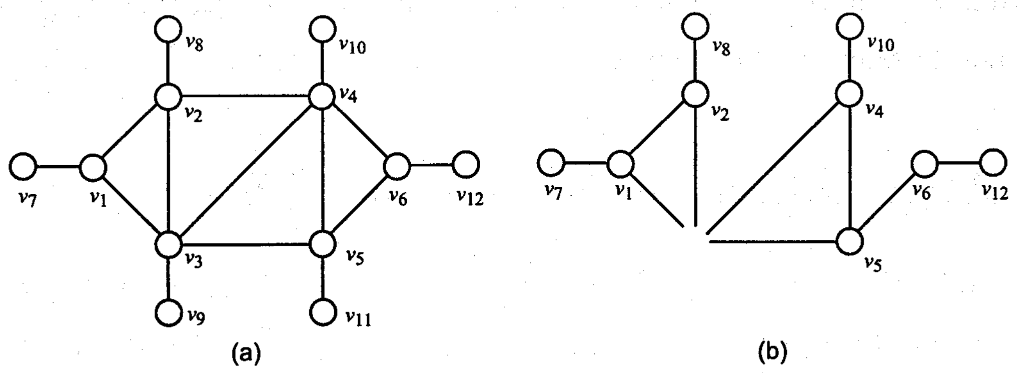

Figure 6.2: Augmented graph AG.

Example 16 Consider the network discussed in the previous example. Figure 6.2(a) shows the augmented graph AG corresponding to this network. When the status of the network is the same as that illustrated in Figure 6.1, the current configuration of AG is $(\{v_1, v_2, v_4, v_5, v_6, v_7, v_8, v_{10}, v_{12}\}, \{e_{1,2}, e_{1,3}, e_{2,3}, e_{3,4}, e_{3,5}, e_{4,5}, e_{5,6}, e_{1,7}, e_{2,8}, e_{4,10}, e_{6,12}\})$. Figure 6.2(b) depicts this configuration.

Let u_i be the event that vertex v_i is up and let $u_{i,j}$ be the event that edge $e_{i,j}$ is up, respectively. Then, the probabilities of these events are given as follows:

$$\begin{array}{lrl}
\text{For} & v_i \in V, & \Pr(u_i) = p_i + q_i \\
\text{For} & v_i \in AV - V, & \left\{ \begin{array}{rcl} \Pr(u_i | u_{i-n}) & = & \frac{p_i}{p_i+q_i} \\ \Pr(u_i | \overline{u_{i-n}}) & = & 0 \end{array} \right. \\
\text{For} & e_{i,j} \in E, & \Pr(u_{i,j}) = p_{i,j} \\
\text{For} & e_{i,j} \in AE - E, & \left\{ \begin{array}{rcl} \Pr(u_{i,j} | u_i) & = & 1 \\ \Pr(u_{i,j} | \overline{u_i}) & = & 0 \end{array} \right.
\end{array}$$

6.4 Computing $h(N)$'s under the Extended Failure Model

Let S be the maximal subgraph embedded in the current configuration (AV', AE'), that is, $S = (SV, SE)$ where $SV = AV'$ and SE is the set of edges in AE' whose end nodes are both in AV'. Let $CC_1, CC_2, \cdots \in SV$ be the connected components of S. It is then clear that $\{v_{i-n} | v_i \in CC_j - V\}$ for each CC_j is a partition group in the current network status unless it is an empty set. For example, if the current configuration of AG is as shown in Figure 6.2(b), S is $(\{v_1, v_2, v_4, v_5, v_6, v_7, v_8, v_{10}, v_{12}\}, \{e_{1,2}, e_{4,5}, e_{5,6}, e_{1,7}, e_{2,8}, e_{4,10}, e_{6,12}\})$. All connected components of S are $\{v_1, v_2, v_7, v_8\}$ and $\{v_4, v_5, v_6, v_{10}, v_{12}\}$, and all partition groups are $\{v_1, v_2\}$ and $\{v_4, v_6\}$.

This fact allows us to adopt the algorithm shown in Figure 4.2, which computes the probability that each node group becomes a partition group under the two-mode failure model, to the extended failure model with only slight modification.

From the discussion in 4.5, it is seen that the event that the maximal subgraph in the current configuration becomes (SV, SE) is represented by the following Boolean expression:

$$\bigwedge_{v_i \in SV} u_i \wedge \bigwedge_{v_i \in AV-SV} \overline{u}_i \wedge \bigwedge_{e_{i,j} \in SE} u_{i,j} \wedge \bigwedge_{e_{i,j} \in AE_{SV}-SE} \overline{u}_{i,j}$$

where AE_{SV} is the set of edges in AE whose end nodes are both in SV.

Let $\hat{prob}(SV, SE)$ be the probability that the maximal subgraph in the current configuration becomes (SV, SE). If there is neither vertex $v_i \in SV - V$ such that $v_{i-n} \notin SV$ nor edge $e_{i,j} \in SE - E$ such that $v_i \notin SV$, this probability is

$$\begin{aligned} \hat{prob}(SV, SE) \; = \; & \prod_{v_i \in SV \cap V} p_i \prod_{v_i \in SV - V} \frac{p_i}{p_i + q_i} \prod_{v_i \in (AV-SV) \cap V} (1 - p_i) \prod_{v_i \in AV-SV-V} (1 - \frac{p_i}{p_i + q_i}) \\ & \prod_{e_{i,j} \in SE \cap E} p_{i,j} \prod_{e_{i,j} \in (AE_{SV}-SE) \cap E} (1 - p_{i,j}). \end{aligned}$$

Otherwise, $\hat{prob}(SV, SE) = 0$.

Based on the above equation, we can compute, by an algorithm similar to the one shown in Figure 4.2, the probability that each node group becomes a partition group. Figure 6.3 shows the formal description of this algorithm. For every subgraph $S = (SV, SE)$ of AG, the proposed algorithm determines all connected components of S, and computes all partition groups from these node groups. Then it calculates $\hat{prob}(SV, SE)$ and accumulates it for the partition groups. Once $h(N)$ is obtained for all N, a coterie with the maximum availability can be obtained using the 0-1 integer programming approach described in Chapter 4.

```
/* Step 1: Initialization */
  For (every node group $N$ of $V$) do
    $h(N) := 0$
  END_For
/* Step 2: Accumulation of $\hat{prob}(SV, SE)$ */
  For (every subgraph $(SV, SE)$ of $AG$) do
    If $v_i \in SV - V$ such that $v_{i-n} \notin SV$ or $e_{i,j} \in SE - E$ such that $v_i \notin SV$ exist,
    then go to the next iteration.
    /* Substep 2.1 Determination of Partition Groups in $(SV, SE)$ */
      $N' := SV, Set\_CCs := \emptyset$
      While ($N'$ is not empty) do
        Select one node $v_i$ from $N'$
        $CC := \{v_i\}$, $N' := N - \{v_i\}$
        $AN := \{v_j \in N' | v_j$ is adjacent to $v_i\}$
        While ($AN$ is not empty) do
          $CC := CC \cup AN$, $N' := N' - AN$
          $AN := \{v_j \in N' | v_j$ is adjacent to a node in $CC\}$
        END_While
        $Set\_CCs := Set\_CCs + CC$
      END_While
      $Set\_PGs := \emptyset$
      While ($Set\_CCs$ is not empty) do
        Select one node group $CC$ from $Set\_CCs$
        $Set\_CCs := Set\_CCs - \{CC\}$
        $PG := \{v_{i-n} | v_i \in CC - V\}$
      END_While
    /* Substep 2.2 Computation and Accumulation of $\hat{prob}(SV, SE)$ */
      If ($PG$ is not empty) Then
        Compute $\hat{prob}(SV, SE)$
        For (every node group $N$ in $Set\_PGs$) do
          $h(N) := h(N) + \hat{prob}$
        END_For
      End_If
  END_For
```

Figure 6.3: Algorithm for computing $h(N)$'s under the three-model failure model.

6.5 Computing Availability under the Extended Failure Model

The evaluation methods for the availability of coteries and the site resiliency of wr-coteries can also be adapted to the new failure model with slight modification. As stated in Chapter 5, the evaluation methods are conceptually similar, so we only discuss how to evaluate the availability of coteries under the new failure model in this chapter.

By definition, it is clear that on the augmented graph AG, if there is a path consisting of up vertices and up edges between $v_i (i > n)$ and $v_j (j > n)$, then two nodes v_{i-n} and v_{j-n} can communicate with each other. For example, consider the case where the status of the network corresponds to the configuration in Figure 6.2(b). In this case, there is a path between v_{10} and v_{12}, so it is seen that nodes v_4 and v_6 can communicate with each other in spite of failures.

Now let $\mathcal{C}$ be a given coterie and let $A(Q)(Q \in \mathcal{C})$ denote $\{v_{i+n} | v_i \in Q\}$. As discussed in Chapter 5, there are some nodes that can enter the critical section in spite of failures, if and only if at least one quorum whose members can communicate with each other exists. This means that the critical section can be entered if and only if there is a connected subgraph of AG that consists of up vertices and edges and spans all vertices in $A(Q)$ for at least one quorum $Q \in \mathcal{C}$.

Based on this fact, we now define an MQST under the three-failure model as a tree of AG that contains all vertices of $A(Q)$ for at least one quorum $Q \in \mathcal{C}$. It is then clear that the critical section can be entered, if and only if for at least one MQST all of its vertices and edges are up. Therefore, the availability of a coterie under the three-mode failure model can be computed in the same manner as the method we proposed for the two-mode failure model. That is, it can be computed by (1) enumerating all MQSTs on AG, and then (2) calculating the probability that at least one MQST consists of up vertices and up edges.

Figure 6.4 shows an algorithm for enumerating MQSTs on AG. This algorithm is the same as the original algorithm shown in Figure 5.1, except that it does not examine trees consisting of one vertex but starts with tree $(\{v_i, v_{i+n}\}, \{e_{i,i+n}\})$ for $i = 1, 2, \cdots, n$. Once all MQSTs are found, availability can be computed using a terminal reliability algorithm, such as [25] or [58].

```
/* Step 1: Initialization */
  $TRY := \{(\{v_1, v_{n+1}\}, \{e_{1,n+1}\}), \cdots, (\{v_n, v_{n+1}\}, \{e_{n,2n}\}\}$
  $FOUND := \emptyset$
/* Step 2: Generation of all MQSTs */
  While ($TRY \neq \emptyset$) do
    /* 2.1 Checking Step */
      For all $t \in TRY$ do
        If ($t$ spans all vertices of $A(Q)$ for at least one quorum $Q$ in the given coterie $\mathcal{C}$) Then
          If (any tree in $FOUND$ is not a subgraph of $t$)
            Then
              add $t$ to $FOUND$
              remove $t$ from $TRY$
            Else
              remove $t$ from $TRY$
          END_If
        END_If
      END_For
    /* 2.2 Expanding Step */
      $NEW := \emptyset$
      For all $t = (V', E') \in TRY$ do
        $IE$ := set of all incident edges on $t$
        $IE' := \{e_{a,b} \in AE | (v_a \in V' \wedge v_b \notin V') \vee (v_b \in V' \wedge v_a \notin V')\}$
        For all ($e_{a,b} \in IE'$) do
          $newt := (V' \cup \{v_a, v_b\}, E' \cup \{e_{a,b}\})$
          add $newt$ to $NEW$
        END_For
      END_For
      $TRY := NEW$
  END_While
```

Figure 6.4: Algorithm for MQST enumeration under the three-mode failure model.

Chapter 7

Conclusions

7.1 Achievements

In this dissertation, research on design and evaluation of coteries in networks with arbitrary topology was addressed. First, we focused on design issues, then we discussed evaluation issues.

In the area of design of coteries, we considered two performance measures, and proposed methods for constructing coteries that are optimal in these measures. The first measure is max-delay, that is, the maximum communication delay needed for delivering a message between a node wishing to enter the critical section and the members of a quorum. The second measure we discussed is availability, which is the probability that the critical section can be entered in spite of failures. Both the problem of finding the coteries with optimal max-delay and the problem of finding those with optimal availability had been solved only for the case where the system has special kinds of topologies, and for general networks, these problems had been left open.

We proposed an algorithm to find max-delay optimal coteries for systems with arbitrary topology, and showed that its time complexity is $O(n^3 \log n)$ where n is the number of nodes.

In addition, we proposed a method based on 0-1 integer programming for constructing coteries with optimal availability. This is an extension of an approach previously proposed by Tang and Natarajan in [62]. A major weakness of the original approach was that there had been no analytic method for calculating the values of some parameters used for the problem formulation. By developing a new algorithm that computes these parameters, we coped with this weakness and succeeded in obtaining optimal coteries for various networks. To the best of our knowledge, this is the first time that optimal coteries have been obtained for general networks with unreliable nodes and links.

As for evaluation issues, we proposed graph-theoretic methods for evaluating dependability-related measures. The first method we proposed was for evaluating the availability of coteries. We introduced a new notion called a Minimal Quorum Spanning Tree (MQST) and developed the evaluation method based on this notion.

In addition, we proposed a method for evaluating the *site resiliency* of wr-coteries. This method is conceptually similar to the availability evaluation method for coteries. We also demonstrated the use of the evaluation method for optimizing the dependability of a well-known replication control scheme called the weighted voting scheme.

Moreover, we proposed a new failure model that considers the status of nodes in more detail, and showed a way to adapt the proposed techniques to this extended model.

7.2 Future Research

There are several directions in which further work is needed. As for minimizing communication delays, we have left open the problem of finding mean-delay optimal coteries. We also leave open the problem of finding delay-optimal coteries when the network reliability is not high, so that operations blocked by failures should be considered.

The problem of constructing wr-coteries with maximum availability is also left open. Though in [62] a formulation of this problem into a 0-1 integer programming problem was proposed, this formulation imposes a restriction that any read quorum is a subset of a write quorum. It is an open problem whether there are situations in which no optimal wr-coterie satisfies this restriction. This is of interest since if there always exists an optimal wr-coterie that satisfies this restriction, we will be able to obtain it using 0-1 integer programming.

Concerning evaluation issues, we think that there is room for improvement of the proposed evaluation methods. These methods are conceptually simple since they consist of two phases. However, a large number of intermediate trees have to be examined at the first phase in order to produce inputs to the second phase. If we could develop a one-phase algorithm that computes availability or site-resiliency directly from a given network, it would work much faster than the proposed methods. In fact, such refinement can be seen in the case of the algorithms for computing *distributed program reliability*. Though the first algorithm consists of two phases[34], several one-phase algorithms were developed after the original algorithm had been proposed (e.g.,[12]).

Considering more severe failures is a problem that deserves further study. For example, malicious failures cannot be tolerated by using a coterie because two quorums in a coterie may have only one node in common, and if this node behaves maliciously, mutual exclusion is no longer achieved. Recently, a notion called a *Byzantine quorum system* has been proposed by Malkhi et al[38]. In a Byzantine quorum system, any pair of quorums shares more than one node for tolerating arbitrary failures. On the other hand, some researchers proposed variants of coteries in which requirement of the intersection property is loosened to some extent (e.g, [31, 39]). These new notions are used for achieving higher concurrency at the cost of weaker consistency requirement. Optimizing the availability of such variants of coteries has not been studied sufficiently, thus it should be explored in the future.

Throughout this dissertation, we have assumed that a coterie does not change once it has been determined. On the contrary, some researchers proposed adaptive schemes that reconfigure coteries dynamically according to the change of the system state (e.g.,[9, 27, 30]). Though such dynamic schemes need additional protocols for detecting failures and reconfiguring coteries, and therefore tend to be complex, they may be preferred in terms of dependability. Design and evaluation of such dynamic schemes also deserve further study.

Bibliography

[1] D.Agrawal and A.El Abbadi, "An efficient and fault-tolerant solution for distributed mutual exclusion," *ACM Trans. Computer Systems*, vol.9, no.1, pp.1-20, 1991.

[2] D.Agrawal and A.El Abbadi, "The generalized tree quorum protocol: an efficient approach for managinh replicated data," *ACM Trans. Database Systems*, vol.17, no.4, pp.689-717, 1992.

[3] D.Agrawal, Ö.Eğecioğlu, and A.El Abbadi, "Billiard quorums on the grid," *Information Processing Letters*, vol.64, no.1, pp.9-16, 1997.

[4] M.Ahamad, M.H.Ammar, and S.Y.Cheung, "Multidimensional voting," *ACM Trans. Computer Systems*, vol.9, no.4, pp.399-431, 1991.

[5] Y.Amir and A.Wool, "Evaluating quorum systems over the Internet," *Proc. of 26th International Symposium on Fault-Tolerant Computing (FTCS-26)*, pp.26-35, 1996.

[6] M.O.Ball, "Computational complexity of network reliability analysis: an overview," *IEEE Trans. Reliability*, vol.35, no.3, pp.230-239, 1986.

[7] D.Barbara and H.Garcia-Molina, "The vulnerability of voting assignments," *ACM Trans. Computer Systems*, vol.4, no.3, pp.187-213, 1986.

[8] D.Barbara and H.Garcia-Molina, "The reliability of voting mechanism," *IEEE Trans. Computers*, vol.36, no.10, pp.1197-1207, 1987.

[9] M.Bearden and R.P.Bianchini, "A fault-tolerant algorithm for decentralized on-line quorum adaptation," *Proc. of 28th International Symposium on Fault-Tolerant Computing (FTCS-28)*, pp.262-271, 1998.

[10] J.C.Bioch and T.Ibaraki, "Generating and approximating nondominated coteries," *IEEE Trans. Parallel and Distributed Systems*, vol.6, no.9, pp.905-914, Sept. 1995.

[11] G.Cao, M.Singhal, Y.Deng, N.Rishe, and W.Sun, "A delay-optimal quorum-based mutual exclusion scheme with fault-tolerance capability," *Proc. of the 18th Intl. Conf. on Distributed Computing Systems (ICDCS'98)*, pp.444-451, 1998.

[12] D.J.Chen and T.H.Huang, "Reliability analysis of distributed systems based on a fast reliability algorithm," *IEEE Trans. Parallel and Distributed Systems*, vol.3, no.2, pp.139-154, 1992.

[13] I.R.Chen and D.C.Wang, "Analysis of replicated data with repair dependency," *The Computer Journal*, vol.39, no.9, pp.767-779, 1996.

[14] Y.G.Chen and M.C.Yuang, "A cut-based method for terminal-pair reliability," *IEEE Trans. Reliability*, vol.45, no.3, pp.413-416, 1996.

[15] S.Y.Cheung, M.Ahamad, and M.H.Ammar, "Optimizing vote and quorum assignments for reading and writing replicated data," *IEEE Trans. Knowledge and Data Engineering*, vol.1, no.3, pp.387-397, 1989.

[16] S.Y.Cheung, M.H.Ammar, and M.Ahamad, "The grid protocol: a high performance scheme for maintaining replicated data," *IEEE Trans. Knowledge and Data Engineering*, vol.4, no.6, pp.582-592, 1992.

[17] S.B.Davidson, H.Garcia-Molina, and D.Skeen, "Consistency in partitioned networks," *ACM Computing Surveys*, vol.17, no.3, pp.341-370, 1985.

[18] K.Diks, E.Kranakis, D.Krizanc, B.Mans, and A.Pelc, "Optimal coteries and voting schemes," *Information Processing Letters*, vol.51, no.1, pp.1-6, 1994.

[19] A.W.Fu, "Delay-optimal quorum consensus for distributed systems," *IEEE Trans. Parallel and Distributed Systems*, vol.8, no.1, pp.59-69, Jan. 1997.

[20] R.W.Floyd, "Algorithm 97: shortest path," *Comm. ACM*, vol.5, no.6, p.345, 1962.

[21] H.Garcia-Molina and D.Barbara, "How to assign votes in a distributed system," *J. ACM*, vol.32, no.4, pp.841-860, 1985.

[22] D.K.Gifford, "Weighted voting for replicated data," *Proc. 7th Symp. Operating System Principles*, pp.150-159, 1979.

[23] T.Harada and M.Yamashita, "Nondominated coteries on graphs," *IEEE Trans. Parallel and Distributed Systems*, vol.8, no.6, pp.667-672, June 1997.

[24] S.Hariri and H.Mutlu, "Hierarchical modeling of availability in distributed systems," *IEEE Trans. Software Engineering*, vol.21, no.1, pp.50-56, 1995.

[25] S.Hariri and C.S.Raghavendra, "SYREL: A symbolic reliability algorithm based on path and cutset methods," *IEEE Trans. Computer*, vol.36, no.10, pp.1224-1232, 1987.

[26] A.A.Helal, A.A.Heddaya, and B.B.Bhargava, *Replication Techniques in Distributed Systems*, Kluwer Academic Publishers, 1996.

[27] M.Herlihy, "Dynamic quorum adjustment for partitioned data," *ACM Trans. on Database Systems,* vol.12, no.2, pp.170-194, 1987.

[28] T.Ibaraki and T.Kameda, "A theory of coteries: mutual exclusion in distributed systems," *IEEE Trans. Parallel and Distributed Systems*, vol.4, no.7, pp.779-794, 1993.

[29] T.Ibaraki, H.Nagamochi, and T.Kameda, "Optimal coteries for rings and related networks," *Distributed Computing*, vol.8, no.4, pp.191-201, 1995.

[30] S.Jajodia and D.Mutchler, "Dynamic voting algorithms for maintaining the consistency of a replicated database," *ACM Trans. on Database Systems,* vol.15, no.2, pp.230-280, 1990.

[31] H.Kakugawa, S.Fujita, M.Yamashita, and T.Ae, "Availability of k-coterie," *IEEE Trans. Computer*, vol.42, no.5, pp.553-558, 1993.

[32] A.Kumar, "Hierarchical quorum consensus: a new algorithm for managing replicated data," *IEEE Trans. Computer*, vol.40, no.9, pp.996-1004, 1991.

[33] A.Kumar and A.Segev, "Cost and availability tradeoffs in replicated data concurrency control," *ACM Trans. Database Systems*, vol.18, no.1, pp.102-131, 1993.

[34] V.K.P.Kumar, S.Hariri, and C.S.Raghavendra, "Distributed program reliability analysis," *IEEE Trans. Software Engineering*, vol.12, no.1, pp.42-50, 1986.

[35] Y.-C.Kuo and S.-T.Huang, "A geometric approach for constructing coteries and k-coteries," *IEEE Trans. Parallel and Distributed Systems*, vol.8, no.4, 1997.

[36] N.A.Lynch and A.A.Shavartsman, "Robust emulation of shared memory using dynamic quorum-acknowledged broadcasts," *Proc. of 28th International Symposium on Fault-Tolerant Computing (FTCS-27)*, pp.272-281, 1997.

[37] M.Maekawa, "A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems," *ACM Trans. Computer Systems*, vol.3, no.2, pp.145-159, 1985.

[38] D.Malkhi and M.Reiter, "Byzantine quorum systems," *Proc. of the 29th ACM Symposium on Theory of Computing*, pp.569-578, 1997.

[39] D.Malkhi, M.Reiter, and R.Wright, "Probabilistic quorum systems," *Proc. of the 16th ACM symposium on Principles of Distributed Computing*, pp.267-273, 1997.

[40] D.A.Menascé, Y.Yesha, and K.Kalpakis, "On a unified framework for the evaluation of distributed quorum attainment protocols," *IEEE Trans. Software Engineering*, vol.20, no.11, pp.868-884, 1994.

[41] A.Moffat and T.Takaoka, "An all pairs shortest path algorithm with expected running time $O(n^2 \log n)$," *SIAM Journal on Computing*, vol.16, no.6, pp.1023-1031, 1987.

[42] S.J.Mullender and P.M.B.Vitanyi, "Distributed match-making," *Algorithmica*, vol.3, pp.367-391, 1988.

[43] S.Nishio, K.Li, and E.G.Manning, "A resilient mutual exclusion algorithm for computer networks," *IEEE Trans. Parallel and Distributed Systems*, no.1, vol.3, pp.344-355, 1990.

[44] D.Peleg and A.Wool, "The availability of quorum systems," *Information and Computation*, vol.123, no.2, pp.210-223, 1995.

[45] D.Peleg and A.Wool, "Crumbling walls: A class of practical and efficient quorum systems," *Distributed Computing*, vol.10, no.2, pp.87-97, 1997.

[46] T.Politof and A.Satyanarayana, "Efficient algorithms for reliability analysis of plannar networks - a survey," *IEEE Trans. Reliability*, vol.35, no.3, pp.252-259, 1986.

[47] S.Rai and D.P.Agrawal, *Distributed Computing Network Reliability,* IEEE Computer Society, 1990.

[48] K.Raymond, "A tree-based algorithm for distributed mutual exclusion algorithm," *ACM Trans. Computer Systems*, vol.7, no.1, pp.61-77, 1989.

[49] S.Rangarajan, S.Setia, and S.K.Tripathi, "A fault-tolerant algorithm for replicated data management," *IEEE Trans. Parallel and Distributed Systems*, vol.6, no.12, pp.1271-1282, 1995.

[50] G.Ricart and A.K.Agrawala, "An optimal algorithm for mutual exclusion in computer networks," *Comm. ACM*, vol.24, no.1, pp.9-17, 1981

[51] A.Rosenthal, "Computing the reliability of a complex network," *SIAM Journal on Applied Mathematics*, vol.32, pp.384-393, 1977.

[52] D.Saha, S.Rangarajan, and S.K.Tripathi, "An analysis of the average message overhead in replica control protocols," *IEEE Trans. Parallel and Distributed Systems*, vol.7, no.10, pp.1026-1034, 1996.

[53] B.Sanders, "The information structure of distributed mutual exclusion algorithm," *ACM Trans. Computer Systems*, vol.5, no.3, pp.284-299, 1987.

[54] A.Satanarayana and R.K.Wood, "A linear-time algorithm for computing K-terminal reliability in series-parallel networks," *SIAM Journal of Computing*, vol.14, no.4, pp.818-832, 1985.

[55] M.Singhal, "A taxonomy of distributed mutual exclusion," *J. Parallel and Distributed Computing*, vol.15, no.1, pp.94-101, 1993.

[56] M.Singhal and N.G.Shivaratri, *Advanced Concepts in Operating Systems,* McGraw-Hill, 1994.

[57] W.Smith and P.Decitre, "An evaluation method for analysis of the weighted voting algorithm for maintaining replicated data," in *Proc. 4th Int. Conf. Distributed Comput. Syst.,* pp.494-502, 1984.

[58] S.Soh and S.Rai, "CAREL: Computer aided reliability evaluator for distributed computing networks," *IEEE Trans. Parallel and Distributed Systems*, vol.2, no.2, pp.199-213, 1991.

[59] M.Spasojevic and P.Berman, "Voting as the optimal static pessimistic scheme for managing replicated data," *IEEE Trans. Parallel and Distributed Systems*, vol.5, no.1, pp.64-73, 1994.

[60] I.Suzuki and T.Kasami, "A distributed mutual exclusion algorithm," *ACM Trans. Computer Systems*, vol.3, no.4, pp.344-349, 1985.

[61] J.Tang, "On multilevel voting," *Distributed Computing*, vol.8, no.1, pp.39-58, 1994.

[62] J.Tang and N.Natarajan, "Obtaining coteries that optimize the availability of replicated databases," *IEEE Trans. Knowledge and Data Engineering*, vol.5, no.2, pp.309-321, 1993.

[63] R.H.Thomas, "A majority consensus approach to concurrency control," *ACM Trans. Database Systems*, vol.4, no.2, pp.180-209, 1979.

[64] Z.Tong and R.Y.Kain, "Vote assignments in weighted voting mechanisms," *IEEE Trans. Computers*, vol.40, no.5, pp.664-667, 1991.

[65] T.W.Yan, H.Garcia-Molina: "Distributed selective dissemination of information," *Proc. of 3rd Inter. Conf. Parallel and Distributed Information Systems (PDIS)*, pp.89-98, 1994.

[66] D.Venkaiah and P.Jalote, "An integer programming approach for assigning votes in a distributed systems," *Proc. of 14th Symposium of Reliable Distributed Systems*, pp.128-134, 1995.