Structure-Based Resilience Metrics for Service-Oriented Networks

DANIEL J. ROSENKRANTZ [1] SANJAY GOEL [2] S. S. RAVI [1] JAGDISH GANGOLLY [3]

October 11, 2004

Abstract

Many governmental agencies and businesses organizations use networked systems to provide a number of services. Such a service-oriented network can be implemented as an overlay on top of the physical network. It is well recognized that the performance of many of the networked computer systems is severely degraded under node and edge failures. The focus of our work is on the resilience of service-oriented networks. We develop a graph theoretic model for service-oriented networks. Using this model, we propose metrics that quantify the resilience of such networks under node and edge failures. These metrics are based on the topological structure of the network and the manner in which services are distributed over the network. Based on this framework, we address two types of problems. The first type involves the analysis of a given network to determine its resilience parameters. The second type involves the design of networks with a given degree of resilience. We present efficient algorithms for both types of problems. Our approach for solving analysis problems relies on known algorithms for computing minimum cuts in graphs. Our algorithms for the design problem are based on a careful analysis of the decomposition of the given graph into appropriate types of connected components.

Keywords: Service-oriented network, resilience, node and edge failure, analysis and design problems, polynomial time algorithms.

Category: Regular Paper.

Contact Author Information:

Name & Address	:	S. S. Ravi Department of Computer Science, LI 67A University at Albany – State University of New York Albany, NY 12222, USA
Email	:	ravi@cs.albany.edu
Phone	:	(518) 442-4278
Fax	:	(518) 442-5638

[1]Department of Computer Science, University at Albany – State University of New York, Albany, NY 12222. Email: {djr,ravi}@cs.albany.edu.

[2]Department of Management Science and Information Systems, School of Business, University at Albany – State University of New York, Albany, NY 12222. Email: goel@albany.edu.

[3]Department of Accounting and Law, School of Business, University at Albany – State University of New York, Albany, NY 12222. Email: j.gangolly@albany.edu.

1 Introduction

1.1 Motivation

Federal and state governmental agencies, businesses and other organizations use networked systems to provide a number of services. Private networks maintained by organizations allow one component of the organization (such as the Emergency Management Agency) to access services provided by other components (such as Transportation Agency, Health Services, etc.). In these situations, fast and reliable access to information is needed. Several papers in the literature have proposed architectures for **service-oriented networks** (see for example [DME02a, DME02b, DMR03, GBI03]). In such an architecture, each node is associated with two sets of services, namely local and non-local services. When a user requests a service that is locally available at a node, the node provides the service directly. When the requested service is non-local, the node forwards the user's request to another node in the network where the service is available, and relays the response to the user. Different approaches for implementing service-oriented networks are available in the literature (e.g. [CZ+99, GJ99, IF01, DMR03, GBI03, GTS03, MKG03, SP03]). Typically, such networks are implemented as overlay networks on top of the application layer.

It is well recognized that many of the networked computer systems used by various government agencies and organizations are not resilient enough to withstand failures and attacks. The performance of these networks is severely degraded by failures. Thus, it is important to develop techniques for designing and implementing resilient service-oriented networks which can not only survive attacks and failures but also continue to provide a reasonable level of service to support critical infrastructure and activities of governmental agencies and organizations. Methods for implementing networks that can continue to function under node and link failures are known (see for example [He99, GJ99, MKG03]). Also, techniques for data replication to ensure availability of data in the event of network failures are available (see for example [CMN03, GL03, DW01, Si01, He99]). In the context of resilient service-oriented networks, services must also be replicated. Further, when a failure or anomalous behavior is sensed by the network, it must have the ability to migrate services to other nodes across the network, taking into consideration the resource constraints at the nodes along the migration path. Such a feature is necessary to allow critical services to be maintained on the network during duress and is especially useful for state and federal agencies in handling crisis situations. Thus, effective methods for the design and analysis of resilient service-oriented networks must accommodate a variety of features and resource constraints.

1.2 Our Contributions

Much of the work in the literature on service-oriented and overlay networks has addressed issues such as the maintenance of routing tables at nodes and reliable transmission of service requests

and network status information across the network. Our work considers the resilience of service-oriented networks at a higher level of abstraction, assuming that lower level mechanisms for basic network functions such as routing and service discovery are supported by the system. The focus of our work is on the identification of suitable metrics that can be used to quantify the resilience of service-oriented networks under node and edge failures. Such metrics are useful in assessing the reliability of a given network (i.e., in analyzing a given network) as well as in choosing an appropriate network topology and/or an optimal distribution of services over the network (i.e., in designing the network).

We propose a graph theoretic model for service-oriented networks and use this model to identify some resilience metrics. These metrics are based on the topology of the network and the manner in which services are distributed over the network. An example of such a metric is **node resilience**, which specifies the maximum number of node failures that a service-oriented network can tolerate and still continue to provide services to users. (Precise definitions of this and other metrics are given in Section 2.) These metrics are different from other notions of network resilience (e.g. the minimum number of nodes or edges whose failure will disconnect the network) studied in the literature (see for example [Pr86, Co87, NG90, HH93, Hw94, Ja94, MOY97]). In particular, our metrics for service-oriented networks explicitly consider the distribution of services over the nodes of a network. Additional discussion regarding these metrics is provided in Section 2.

Having identified some resilience metrics, we develop algorithms for analysis and design problems arising in the context of resilient service-oriented networks. Given a service-oriented network, the goal of the analysis problem is to compute the node and edge resilience parameters of the network. We develop polynomial time algorithms for these problems. These algorithms are derived through a transformation to the problem of computing minimal cutsets in graphs. The design problem addressed in this paper concerns the placement of services at the nodes of a given network so that the cost of placing the services is minimized and the resulting network has a specified level of resilience. We consider this problem for single node and single edge failures, and develop polynomial algorithms. These algorithms are based on a careful analysis of the decomposition of the given graph into appropriate types of connected components.

1.3 Organization of the Paper

The remainder of this paper is organized as follows. Section 2 presents our formal model and the definitions of the resilience metrics considered in this paper. Section 3 presents algorithms for analyzing a given service-oriented network. Section 4 presents algorithms for designing service-oriented networks with a given degree of resilience. Section 5 provides some concluding remarks and directions for further research.

2 Formal Model and Structure-Based Resilience Metrics

2.1 Graph Model and Definitions of Metrics

Following standard practice [Ja94, Pr86], we model a network as an undirected connected graph. Each node represents a computer system and each edge represents a bidirectional link between the corresponding pair of systems. To model service-oriented networks, we associate two sets of services with each node. For a node v, $A(v)$ denotes the set of services available locally at v, and $N(v)$ denotes the set of nonlocal services needed at v. In other words, node v supports each service in $N(v)$ by forwarding requests for such a service to one or more nodes that offers it as a local service. Thus, the sets $A(v)$ and $N(v)$ are disjoint. A user connected to node v may request any service in $A(v) \cup N(v)$. For a node v, if service $s \in N(v)$, we say that v is a **demand point** for service s.

The nodes and/or links of a network may fail either because of an attack or because of equipment failure. We assume that node failures corresponds to system crashes. Thus, we don't consider Byzantine node failures. In general, a system that has crashed cannot communicate with any of its neighbors. Thus, each node failure can be modeled by deleting the failed node and all the edges incident on the failed node from the underlying graph of the network. When a link fails, it is assumed that no communication across the link is possible in either direction. Thus, each link failure may be modeled by the deletion of the corresponding edge from the underlying graph. Using these models for node and edge failures, several network resilience metrics have been considered in the literature (see for example [Pr86, Co87, NG90, HH93, Hw94, Ja94, MOY97, JW+00, BOS01, LK+03]). Examples of such metrics include node and edge connectivity parameters of a graph (i.e., the minimum number of nodes or edges that must be removed to disconnect the graph). Such metrics characterize the ability of a network to remain connected when nodes and edges fail. They do not take into account the distribution of services across a network. As will be seen, a service-oriented network may continue to function even when the underlying graph is disconnected. Thus, new metrics are needed to capture the notion of resilience in service-oriented networks.

Node and link failures may partition a network into a collection of two or more connected components or subnetworks. In such a case, nodes in one subnetwork cannot access the services provided by the nodes in another subnetwork. We say that a subnetwork is **self-sufficient** if each service needed by a node in that subnetwork is provided by some node in the same subnetwork; otherwise, the subnetwork is said to be **deficient**. Thus, a self-sufficient subnetwork can continue to process requests from users even though it has been sequestered from the rest of the network due to node and link failures. Given a network $G(V, E)$ and a subset $S \subseteq V \cup E$ of nodes and/or edges, we say that the network G is **resilient** with respect to the failure set S if each of the subnetworks that results when all the nodes/edges in S fail is self-sufficient.

We can now define the structure-based resilience metrics proposed in this paper. To do this,

we consider failure sets under two separate categories, namely node failures and edge failures. In the former, each failure set consists only of nodes, and in the latter, each failure set consists only of edges.

Definition 2.1 *(a) A service-oriented network is* k***-edge-failure-resilient*** *if no matter which subset of* k *or fewer edges fails, each resulting subnetwork is self-sufficient. The* ***edge resilience*** *of a network is the largest integer* k *such that the network is* k*-edge-failure-resilient.*

(b) A service-oriented network is k***-node-failure-resilient*** *if no matter which subset of* k *or fewer nodes fails, each resulting subnetwork is self-sufficient. The* ***node resilience*** *of a network is the largest integer* k *such that the network is* k*-node-failure-resilient.*

We now present an example to illustrate the resilience metrics.

Example: Consider the eight node network shown in Figure 1. The eight services provided by the network are denoted by s_1 through s_8. For each node v_i, the figure also shows the sets A_i (the set of services available at v_i) and N_i (the set of services needed at v_i), $1 \leq i \leq 8$. Note that the node connectivity and the edge connectivity of the network are both one, since the network can be disconnected by removing one node (for example, the node v_3) or one edge (the edge $\{v_3, v_5\}$). However, the node and edge resilience parameters of the network are both two. In particular, the subnetworks obtained by deleting the edge $\{v_3, v_5\}$ or one of the nodes v_3 and v_5 are all self-sufficient. It can be verified that no matter which pair of vertices or which pair of edges is deleted, each of the resulting subnetworks is self-sufficient. However, when the three edges $\{v_1, v_2\}$, $\{v_1, v_3\}$ and $\{v_1, v_4\}$ are deleted, the subnetwork containing only the node v_1 is deficient, since it does not have access to service s_4. Likewise, when the three nodes v_1, v_2 and v_3 are deleted, the subnetwork containing only the node v_4 is deficient, since it does not have access to service s_1.

2.2 Problem Formulation

We now provide precise formulations of the analysis and design problems considered in this paper. We start with a specification of analysis problems.

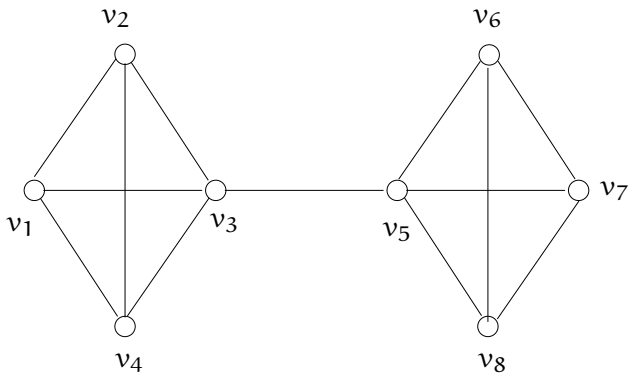
Edge Resilience Problem (ERP)

Instance: A service oriented network consisting of an undirected graph $G(V, E)$ and the sets A_v and N_v for each node $v \in V$.

Requirement: Compute the edge resilience of the network.

The formulation of **Node Resilience Problem** (NRP) is similar to that of ERP. Section 3 presents efficient algorithms for these two problems.

Many versions of design problems for resilient service-oriented networks can be formulated. We focus on design problems where the goal is choose an optimal distribution of services over



$$
\begin{array}{llllll}
A_1 = \{s_1, s_2, s_3\} & & N_1 = \{s_4\} \\
A_2 = \{s_1, s_2, s_4\} & & N_2 = \{s_3\} \\
A_3 = \{s_1, s_3, s_4\} & & N_3 = \{s_2\} \\
A_4 = \{s_2, s_3, s_4\} & & N_4 = \{s_1\} \\
A_5 = \{s_5, s_6, s_7\} & & N_5 = \{s_8\} \\
A_6 = \{s_5, s_6, s_8\} & & N_6 = \{s_7\} \\
A_7 = \{s_5, s_7, s_8\} & & N_7 = \{s_6\} \\
A_8 = \{s_6, s_7, s_8\} & & N_8 = \{s_5\}
\end{array}
$$

Figure 1: Example to Illustrate Resilience Metrics

a given network. In other words, we assume that the network topology and the set of services needed at each node are given, and the objective is to find the set of services to be provided by each node so as to achieve a desired degree of node or edge resilience. In the absence of cost considerations, such problems can be solved trivially by having all the services at each node. In general, such solutions are economically infeasible. Moreover, some of the nodes in the network may not have the computational resources needed to support a certain service. Therefore, we assume that there is a cost associated with placing services at nodes and that the total cost of placing the services must be minimized. We can now provide a precise formulation of the design problems considered in this paper.

Design for Edge Resilience (DER)

Instance: An undirected graph $G(V, E)$, with $V = \{v_1, v_2, \ldots, v_n\}$, a set $S = \{s_1, s_2, \ldots, s_p\}$ of services, a set $N(v_i) \subseteq S$ for each node $v_i \in V$, an $n \times p$ cost matrix $C = [c_{ij}]$, where $c_{ij} \geq 0$ is a real number that denotes the cost of placing service p_j at node v_i, $1 \leq i \leq n$ and $1 \leq j \leq p$, and integer K.

Requirement: For each node $v_i \in V$, compute a set $A(v_i) \subseteq S$ of services to be placed at v_i, such that the edge resilience of the resulting service-oriented network is at least K and the total cost of

placing the services is minimized.

A solution to the DER problem may place a service s_j in at a node v_i when $s_j \in N(v_i)$. In such a case, the sets $A(v_i)$ and $N(v_i)$ for node v_i are no longer disjoint. To ensure that the two sets remain disjoint, one can modify the set $N(v_i)$ into a new need set $N'(v_i)$ by deleting the services in $N(v_i) \cap A(v_i)$.

The formulation of the Design for Node Resilience (DNR) problem is similar, except that the parameter K represents the required level of node resilience.

In Section 4, we present polynomial algorithms for the DER and DNR problems, assuming that the value of K is 1. Even for $K = 1$, the algorithms involve careful analyses of certain types of decompositions of undirected graphs. For larger values of node and edge resilience parameters, developing properties of appropriate graph decompositions appears to be a nontrivial task. So, we leave the design problems for higher resilience values as directions for future work.

3 Algorithms for Analyzing a Given Network

3.1 An Algorithm for Computing Edge Resilience

In this section, we present our algorithm for computing the edge resilience of a given service-oriented network. We begin with some definitions.

Let $G(V, E)$ denote the underlying graph of the given network. We use the term **subnetwork** to mean a connected subgraph of G. (Subnetworks may contain just a single node.) For each node v in a subnetwork H, the sets $A(v)$ and $N(v)$ are the same as those in G. For any subnetwork H and any service $s_j \in S$, we say that H is **deficient with respect to** s_j if there is a node in H that needs s_j and no node in H provides s_j. Thus, a subnetwork H is deficient[4] if there is a service s_j with respect to which H is deficient.

Definition 3.1 *Let $G(V, E)$ denote the underlying graph of a service-oriented network. Let S denote the set of all services available in the network.*

*(a) Given a service $s_j \in S$, a set of edges $Q \subseteq E$ is called a **deficiency inducing edge** (DIE) **set** for G **with respect to service** s_j if at least one of the connected components of the graph $G'(V, E-Q)$ is deficient with respect to s_j.*

*(b) A set of edges $Q \subseteq E$ is a **deficiency inducing edge set** for G if at least one of the connected components of the graph $G'(V, E - Q)$ is deficient.*

A direct consequence of the above definitions is that if Q^* is a DIE set of minimum cardinality for G, then the edge resilience of G is equal to $|Q^*| - 1$. Therefore, we focus on computing $|Q^*|$. Our approach for finding $|Q^*|$ relies on the following observation.

[4]The definition of a deficient subnetwork was presented in Section 2.1.

Observation 3.1 *Let* $G(V, E)$ *denote the underlying graph of a service-oriented network. Let* S *denote the set of all services available in the network. For each service* $s_j \in S$*, let* σ_j *denote the minimum cardinality of a* DIE *set for* G *with respect to service* s_j*. Let* σ^* *denote the minimum cardinality of a* DIE *set for* G*. Then,* $\sigma^* = \min\{\sigma_j : 1 \leq j \leq |S|\}$. ■

The above observation points out that the cardinality of a smallest DIE set for G can be computed by considering each service separately. We now show that for any service s_j, the problem of computing a minimum cardinality DIE set with respect to s_j can be solved by a transformation to the problem of computing minimum weight edge cutsets in undirected graphs. The relevant definitions are given below.

Definition 3.2 *Let* $G(V, E)$ *be an undirected graph with a nonnegative weight* $w(e)$ *for each edge* $e \in E$*. Let* s *and* t *be two distinct vertices in* V*. An* s-t ***edge cutset*** *for* G *is a subset* $E' \subseteq E$ *such that in the graph* $G'(V, E - E')$*, there is no path between* s *and* t*. A* ***minimum weight*** s-t ***edge cutset*** *for* G *is an edge cutset whose total weight is minimum.*

The following well known result shows that minimum weight edge cutsets can be found efficiently (see for example [SW97]).

Theorem 3.1 *Given an undirected graph* $G(V, E)$ *with a nonnegative weight* $w(e)$ *for each edge* $e \in E$ *and two distinct vertices* s *and* t *in* V*, a minimum weight* s-t *edge cutset for* G *can be computed in* $O(|E| + |V| \log |V|)$ *time.* ■

We now explain how an algorithm for the minimum weight s-t edge cutset problem can be used to solve the problem of computing a minimum cardinality DIE set for a given service s_j. We need the following definition.

Definition 3.3 *Let* $G(V, E)$ *be the given service-oriented network and let* s_j *be a given service. Let* $P_j \subseteq V$ *denote the set of nodes that provide service* s_j*. The* ***auxiliary graph*** $G_j(V_j, E_j)$ *for* s_j *is an undirected edge weighted graph constructed as follows.*

(a) The node set $V_j = V \cup \{s\}$*, where* s *is a new node that does not appear in* V*.*

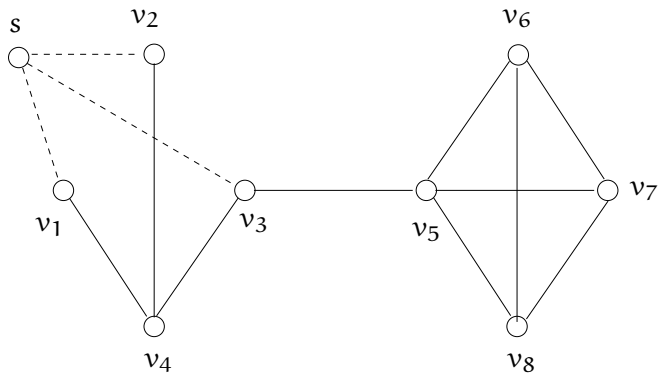
(b) The edge set $E_j = E_j^1 \cup E_j^2$*, where*

(i) $E_j^1 = E - \{\{x, y\} : x \text{ and } y \text{ are both in } P_j\}$ *and*

(ii) $E_j^2 = \{\{s, y\} : y \in P_j\}$.

(c) The weight of each edge in E_j^1 *is 1 and the weight of each edge in* E_j^2 *is* ∞.

As an example, the auxiliary graph of the network of Figure 1 with respect to service s_1 is shown in Figure 2. The usefulness of the auxiliary graph is shown in the following lemma.



Note: Nodes v_1, v_2 and v_3 provide service s_1. Each dotted edge has weight = ∞; other edges have a weight of 1.

Figure 2: Auxiliary Graph with respect to Service s_1 of the Network in Figure 1

Lemma 3.1 *Let* $G(V, E)$ *be the given service-oriented network and let* s_j *be a given service. Suppose* $G_j(V_j, E_j)$ *denotes the auxiliary graph for* s_j *and* v *is a node of* G *that needs service* s_j.

(a) If there is a set $E_v \subseteq E$ *such that there is no path in* $G'(V, E - E_v)$ *between* v *and any node that provides service* s_j*, then* G_j *has an* s-v *edge cutset of weight at most* $|E_v|$.

(b) For any finite integer α*, if* G_j *has an* s-v *edge cutset of weight* α*, then there is a set* $E_v \subseteq E$ *such that* $|E_v| = \alpha$*, and there is no path in* $G'(V, E - E_v)$ *between* v *and any node that provides service* s_j.

Proof:

Part (a): Suppose E_v is a set of edges such that there is no path in the graph $G'(V, E - E_v)$ between v and any node that provides service s_j. Let $E_v^1 \subseteq E_v$ be the set of edges obtained by deleting from E_v each edge $\{x, y\}$ such that both x and y are nodes that provide service s_j. Note that each edge in E_v^1 is also an edge in G_j and the total weight of the edges in E_v^1 is at most $|E_v|$. Using the fact that there is no path in $G'(V, E - E_v)$ between v and any node that provides service s_j, it can be verified that E_v^1 is an s-v edge cutset for G_j.

Part (b): For some finite α, suppose $Q_v \subseteq E_j$ is an s-v edge cutset with weight α for G_j. Since α is finite, Q_v cannot contain any edge incident on node s. Thus, each edge in Q_v is also in $G(V, E)$. Using the fact that Q_v is an s-v edge cutset for G_j, it can be verified that there is no path in $G'(V, E - Q_v)$ between v and any node that provides service s_j. This completes the proof. ■

The following lemma is a direct consequence of Lemma 3.1.

Lemma 3.2 *Let* $G(V, E)$ *be the given service-oriented network and let* s_j *be a given service. Let* $G_j(V_j, E_j)$ *denote the auxiliary graph for* s_j*. For any node* $v \in V$*, the minimum number of edges to be deleted from* G

Input: A service-oriented network $G(V, E)$, the set S of all services, the sets $A(v)$ and $N(v)$ for each node $v \in V$.

Requirement: Find the edge resilience of G.

Algorithm:

1. **for** each service $s_j \in S$ **do**
   (a) Construct auxiliary graph $G_j(V_j, E_j)$ for service s_j.
   (b) Find the set $D_j \subseteq V_j$ of the demand points for service s_j (i.e., the set of nodes that need service s_j).
   (c) **for** each node $v \in D_j$ **do**
   Compute $\alpha_{v,j}$, the minimum weight of an s-v edge cutset in G_j.
   (d) Let $\sigma_j = \min\{\alpha_{v,j} : v \in D_j\}$.
2. Edge resilience of $G = \min\{\sigma_j : s_j \in S\} - 1$.

Figure 3: Algorithm for Computing Edge Resilience

so that there is no path between v and any node that provides service s_j is equal to the weight of a minimum weight s-v edge cutset in G_j. ■

From Lemma 3.2, it follows that the cardinality of a minimum DIE set with respect to a service s_j can be obtained by computing the minimum weight edge cutset in the auxiliary graph G_j for each pair s-v, where v is a node that needs service s_j. Once we find the cardinality of a minimum DIE set with respect to each service s_j, the edge resilience of the given network G can be found by taking the minimum over all services (Lemma 3.1). These observations lead to the algorithm shown in Figure 3 for computing the edge resilience of a given service-oriented network.

We now estimate the running time of the algorithm. Suppose the given network has n nodes, m edges and a total of p services. The running time of the algorithm in Figure 3 is dominated by the minimum weight edge cutset computations. For each service s_j, the algorithm uses $O(n)$ cutset computations. So, the total number of such computations is $O(pn)$. Since each cutset computation can be done in $O(m + n \log n)$ time (Theorem 3.1), the running time of the algorithm is $O(pn(m + n \log n))$. The following theorem summarizes the above discussion.

Theorem 3.2 *Given a service-oriented network $G(V, E)$ and the set of service S, the edge resilience of the network can be computed in $O(pn(m + n \log n))$ time, where $n = |V|$, $m = |E|$ and $p = |S|$.* ■

3.2 An Algorithm for Computing Node Resilience

Our algorithm for computing the node resilience of a service-oriented network follows the same approach as that of edge resilience. The main difference is that we need to work with node cutsets instead of edge cutsets.

When a subset X of nodes is deleted from a graph $G(V, E)$, each edge incident on a node in X is also deleted. Keeping this in mind, it is straightforward to modify the definition of a deficiency inducing edge (DIE) set to obtain the definition of a deficiency inducing node (DIN) set.

Definition 3.4 *Let $G(V, E)$ denote the underlying graph of a service-oriented network. Let S denote the set of all services available in the network. For any subset of nodes X, let $G_X(V - X, E_X)$ denote the subgraph of G obtained by deleting the nodes in X.*

*(a) Given a service $s_j \in S$, a set of nodes $X \subseteq V$ is called a **deficiency inducing node** (DIN) **set** for G **with respect to service** s_j if at least one of the connected components of the graph $G_X(V-X, E_X)$ is deficient with respect to s_j.*

*(b) A set of nodes $X \subseteq E$ is a **deficiency inducing node set** for G if at least one of the connected components of the graph $G_X(V - X, E_X)$ is deficient.*

As in the case of edge resilience, it can be seen that a minimum cardinality DIN set for G can be computed by considering each service separately. To compute that value, we use a transformation to the problem of computing minimum weight node cutsets in graphs. The following definition is the node cutset analog of Definition 3.2.

Definition 3.5 *Let $G(V, E)$ be an undirected graph with a nonnegative weight $w(v)$ for each node $v \in V$. Let s and t be two distinct vertices in V such that $\{s, t\} \notin E$. An s-t **node cutset** for G is a subset $V' \subseteq V - \{s, t\}$ such that when the nodes in V' are deleted from G, there is no path between s and t. A **minimum weight** s-t **node cutset** for G is a node cutset whose total weight is minimum.*

As indicated by the following result from [Ev79], minimum weight node cutsets can be found efficiently.

Theorem 3.3 *Given an undirected graph $G(V, E)$ with a nonnegative weight $w(v)$ for each node $v \in V$ and two distinct vertices s and t in V such that $\{s, t\} \notin E$, a minimum weight s-t node cutset for G can be computed in $O(|E||V|^{1/2})$ time.* ■

The definition of the auxiliary graph used for computing node resilience is the same as that given in Definition 3.3, except that edge weights are not used, and the weight of each node is 1. The usefulness of the auxiliary graph is indicated in the following lemma, whose proof is analogous to that of Lemma 3.2.

Input: A service-oriented network $G(V, E)$, the set S of all services, the sets $A(v)$ and $N(v)$ for each node $v \in V$.

Requirement: Find the node resilience of G.

Algorithm:

1. **for** each service $s_j \in S$ **do**
   (a) Construct auxiliary graph (with node weights instead of edge weights) $G_j(V_j, E_j)$ for service s_j.
   (b) Compute $D_j \subseteq V_j$, the set of demand points for service s_j.
   (c) **for** each node $v \in D_j$ **do**
   Compute $\gamma_{v,j}$, the minimum weight of an s-v node cutset in G_j.
   (d) Let $\Gamma_j = \min\{\gamma_{v,j} : v \in D_j\}$.
2. Node resilience of $G = \min\{\Gamma_j : s_j \in S\} - 1$.

Figure 4: Algorithm for Computing Node Resilience

Lemma 3.3 *Let $G(V, E)$ be the given service-oriented network and let s_j be a given service. Let $G_j(V_j, E_j)$ denote the auxiliary graph (with node weights) for s_j. For any node $v \in V$, the minimum number of nodes to be deleted from G so that there is no path between v and any node that provides service s_j is equal to the weight of a minimum weight s-v node cutset in G_j.* ■

The rest of the computation is similar to that of edge resilience. The resulting algorithm for computing node resilience is shown in Figure 4.

Suppose the given network has n nodes, m edges and a total of p services. The running time of the algorithm in Figure 4 is also dominated by the time for node cutset computations. The total number of node cutset computations is $O(pn)$. Since each cutset computation can be done in $O(mn^{1/2})$ time (Theorem 3.3), the running time of the algorithm is $O(pmn^{3/2})$). The following theorem summarizes the above discussion.

Theorem 3.4 *Given a service-oriented network $G(V, E)$ and the set of service S, the node resilience of the network can be computed in $O(pmn^{3/2})$ time, where $n = |V|$, $m = |E|$ and $p = |S|$.* ■

4 Algorithms for Designing Resilient Networks

4.1 Preliminary Definitions

As mentioned in Section 2.2, the design problem for 1-node or 1-edge resilience assumes that we are given the underlying graph $G(V, E)$, and for each node $v_i \in V$, the set $N(v_i)$ of services needed

at v_i. In addition, a nonnegative cost matrix $C = [c_{ij}]$, where c_{ij} is the cost of placing service s_j at node v_i is also given. The goal is to select a set of services to be placed at each node so that the resulting network has the required level of edge or node resilience, and the total cost of placing the services is minimized.

We saw in Section 3 that the analysis problems for edge and node resilience can be solved by considering each service separately. This idea extends to the design problems as well since the placement of one service has no impact on the placement of other services. So, our approach for solving the design problems also considers one service at a time.

Consider any service s_j. Recall that each node v_i such that $s_j \in N(v_i)$ is a **demand point** for s_j. Each node at which service s_j is placed is called a **service point**. A set of service points for s_j is called a **placement** for s_j. Given a connected graph $G(V, E)$ and a placement P for service s_j, we say that the placement is **1-edge-resilient with respect to service** s_j if for every edge $e \in E$, the graph $G'(V, E - \{e\})$ contains a path from each demand point for s_j to a service point for s_j. The definition of a 1-node-resilient placement can be given in a similar manner. Thus, an equivalent way of posing the design problems is the following: find a placement for each service so that the resulting service-oriented network is 1-edge-resilient (or 1-node-resilient) and the total cost of placement is minimized. This formulation is used in the remainder of this paper.

The next two subsections consider the design problem for 1-edge resilience and 1-node resilience respectively. For reasons of space, we will assume uniform cost values for services in this version; that is, we assume that $c_{ij} = 1$ for all i and j. Our algorithms can be extended to nonuniform cost values. These extensions will be included in a longer version of this paper.

4.2 Designing a 1-Edge-Resilient Network

In this section, we develop our algorithm for the design problem for 1-edge resilience. As mentioned above, each service can be considered separately in solving this problem. So, we will focus our attention on one service, say s_j. We say that a placement P for service s_j is **optimal** if P provides 1-edge-resilience with respect to s_j, and $|P|$ is the smallest among all placements which have the resilience property.

To develop our algorithm for this problem, we recall a standard definition from graph theory [We96].

Definition 4.1 *Let $G(V, E)$ be a connected undirected graph. A **bridge** of G is an edge $\{x, y\}$ whose removal disconnects G. If G has no bridges, then G is called a **bridgeless graph**.*

The following is a well known result in graph theory [We96].

Lemma 4.1 *Let $G(V, E)$ be a connected undirected graph. Suppose G has b bridges given by $E' = \{e_1, e_2, \ldots, e_b\}$. Then, the graph $G'(V, E - E')$ has exactly $b + 1$ connected components and each of these components is a bridgeless graph.* ■

The bridgeless components (BLCs) of the underlying graph $G(V, E)$ play an important role in solving the design problem. To show this connection, we define another auxiliary graph for G as follows.

Definition 4.2 *Let $G(V, E)$ be a connected undirected graph and let $E' = \{e_1, e_2, \ldots, e_b\}$ denote the set of bridges of G. The **BLC graph** of G, denoted by $G_B(V_B, E_B)$, is defined as follows.*

(a) Each node of V_B corresponds to a BLC of G.

(b) For nodes x and y in V_B, the edge $\{x, y\}$ is in E_B if and only if there is a bridge in G that joins a node in the BLC corresponding to x to a node in the BLC corresponding to y.

Intuitively, the BLC graph of G is constructed from G by collapsing each BLC of G into a single super node; the edges of the BLC graph are in one-to-one correspondence with the bridges of G.

Example: The service-oriented network of Figure 1 has one bridge, namely the edge $\{v_3, v_5\}$. One bridgeless component of G is formed by the node set $\{v_1, v_2, v_3, v_4\}$ and the other is formed by $\{v_5, v_6, v_7, v_8\}$. The BLC graph corresponding to this network has two nodes joined by a single edge.

The following is an easy observation concerning the BLC graph of a connected graph G.

Observation 4.1 *Suppose G is a connected undirected graph. The BLC graph of G is a tree.* ■

We need some additional definitions to point out the role played by the BLC graph in solving the 1-edge-resilient design problem. It should be noted that the following definitions are all with respect to the service s_j under consideration.

Definition 4.3 *Let $G(V, E)$ be a connected graph and let G_B denote the BLC graph of G. Let s_j be a service.*

*(a) A **demand component of** G is a BLC of G that has at least one demand point for s_j.*

*(b) The **demand subgraph** G_D^j of G consists of nodes v_H corresponding to the demand components of G, and all edges and nodes of G_B that lie along some path connecting two such nodes.*

Since the BLC graph G_B is a tree (Observation 4.1), the demand subgraph G_D^j is a subtree of G_B. We call each leaf of G_D^j (i.e., a node of degree 1 in G_D^j) a **demand leaf**. Each BLC of G corresponding to a leaf of G_D^j is called a **demand leaf component**.

The importance of demand leaf components of G in finding an optimal placements is shown in the following lemmas.

Lemma 4.2 *Let G be a connected graph. Let G_B denote the BLC graph of G, and let G_D^j denote the demand subgraph of G_B for service s_j. Let δ denote the number of demand leaves of G_D^j. If P^* denote an optimal placement for s_j, then $|P^*| \geq \delta$.*

Proof: If $\delta = 1$, the result follows immediately. For $\delta \geq 2$, we prove the result by contradiction. Suppose $|P^*| < \delta$. For each demand leaf v_H of G_D^j, let $\Pi(v_H)$ denote the set of nodes in the largest subtree of G_B that contains v_H, but does not contain any edges from G_D^j. Since $|P^*| < \delta$, there is a demand leaf v_H of G_D^j such that none of the BLCs corresponding to the nodes in $\Pi(v_H)$ contains a service point for s_j. Since v_H is a leaf of G_D^j, the graph G_D^j contains only one edge e incident on v_H. Let this edge $e = \{v_H, v_X\}$ join node v_H to node v_X. Note that e corresponds to a bridge e' of G. When e' is deleted from G, nodes in the BLC H of G corresponding to node v_H of G_D^j have paths only to nodes in the BLCs corresponding to nodes in $\Pi(v_H)$. Since there are no service points in those BLCs, P^* is not resilient to the failure of e'. This is a contradiction, and Lemma 4.2 follows. ■

Lemma 4.3 *Let G be a connected graph. Let G_B denote the BLC graph of G, and let G_D^j denote the demand subgraph of G_B for service s_j. Let P be a placement for s_j obtained by choosing an arbitrary node from each demand leaf component of G. Then, P is 1-edge-resilient.*

Proof: The chosen placement P has one node from each of the demand leaf component of G. Since every node in G_D^j has a path to a leaf node, G contains a path from each demand point to a service point for s_j. Thus, we can complete the proof of the lemma by showing that this property continues to hold even after deleting any single edge of G.

If G_D^j has only one node, say v_H, then the proof is trivial since the corresponding component H of G is bridgeless. So, for the remainder of this proof, we assume that G_D^j has two or more nodes.

Consider the deletion of any edge e of G. There are three cases to consider.

Case 1: Edge e is not a bridge of G. In this case, deleting e does not disconnect G. Thus, a path from every demand point to a service point for s_j continues to exist even after e is deleted.

Case 2: Edge e is a bridge of G, but does not correspond to an edge of G_D^j. Even in this case, the paths from the demand points to the service points for s_j continue to exist after e is deleted.

Case 3: Edge e is a bridge of G and corresponds to an edge e' of G_D^j. Let X and Y denote the two BLCs of G which are joined by e. Let v_X and v_Y denote the nodes corresponding to X and Y in G_D^j, so that the edge e' in G_D^j joins nodes v_X and v_Y. Since G_D^j is a tree, deleting e' produces only two subtrees. Further, each resulting subtree has at least one leaf node, and the BLC corresponding to each such leaf node has a service point for s_j. Therefore, the deletion of e from G does not create any component that is deficient with respect to s_j. This completes the proof of Lemma 4.3. ■

It follows from Lemmas 4.2 and 4.3 that an optimal placement for service s_j can be found by choosing an arbitrary node from each demand leaf component of G. When this process is repeated for each service, we obtain an optimal placement that achieves 1-edge-resilience. The steps of the resulting algorithm are shown in Figure 5.

We now analyze the running time of the algorithm in Figure 5. As in Section 3, let $|V| = n$, $|E| = m$ and $|S| = p$. Finding the BLCs of G can be done in $O(m + n)$ time [CLRS01]. Then,

Input: A connected graph $G(V, E)$, the set S of all services, the set $N(v_i)$ for each node $v_i \in V$.

Requirement: For each service s_j, find a service point set P_j (i.e., the subset of V at which service s_j will be placed) so that the resulting network is 1-edge-resilient and $|P_j|$ is the smallest among all placements that provide 1-edge-resilience.

Algorithm:

1. Find the bridgeless components (BLCs) of G and construct the BLC graph G_B of G.
2. **for** each service $s_j \in S$ **do**
   (a) Compute the demand point set D_j for s_j.
   (b) Initialize service point set P_j to $\emptyset$.
   (c) Construct the demand subgraph G_D^j and find its leaf nodes.
   (d) **for** each leaf v of G_D^j **do**
   Choose an arbitrary node w from the BLC of G corresponding to v, and add w to P_j.
3. Output the sets P_j, $1 \leq j \leq |S|$.

Figure 5: Algorithm for Designing a 1-Edge-Resilient Network

constructing the BLC graph G_B can be done in $O(n)$ time, since G_B is a tree. The time used to find the placement for each service s_j can be estimated as follows. Assuming that the set $N(v_i)$ for each node v_i is stored as a bit vector of length p, the set D_j of demand points for s_j can be found in $O(n)$ time. Constructing the demand subgraph G_D^j and choosing a node from each demand leaf component of G can also be done in $O(n)$ time. Thus, the time spent in finding a placement for each service is $O(n)$. Hence, the time used to find placements for all services is $O(np)$. Therefore, the overall running time of the algorithm is $O(m + np)$. The following theorem summarizes the main result of this section.

Theorem 4.1 *Given a connected graph $G(V, E)$, the set of services S and the set $N(v_i)$ for each $v_i \in V$, the design problem for 1-edge-resilience under uniform service costs can be solved in $O(m + np)$ time, where $n = |V|$, $m = |E|$ and $p = |S|$.* ■

4.3 Designing a 1-Node-Resilient Network

We now address the problem of computing an optimal placement for achieving 1-node-resilience. The approach is similar to that of the design problem for 1-edge-resilience. We start with some standard graph theoretic definitions [We96].

Definition 4.4 *Let $G(V, E)$ be a connected undirected graph.*

*(a) A node $v \in V$ is a **cut point** (or **articulation point**) if the removal of v disconnects G.*

*(b) A **block** is maximal subgraph G' of G such that G' does not have a cut point.*

Example: Consider the graph $G(V, E)$ shown in Figure 1. It has two cut points, namely nodes v_3 and v_5. G has three blocks: the subgraph induced on the set $\{v_1, v_2, v_3, v_4\}$, the edge $\{v_3, v_5\}$ and the subgraph induced on the set $\{v_5, v_6, v_7, v_8\}$.

Definition 4.5 *Let $G(V, E)$ be a connected undirected graph. The **block-cut point graph (BC graph)** of G, denoted by $G_B(V_B, E_B)$, is the bipartite graph defined as follows.*

(a) V_B has one node corresponding to each block and one node corresponding to each cut point of G.

(b) Each edge $\{x, y\}$ in E_B joins a block node x to a cut point y if the block corresponding to x contains the cut point node corresponding to y.

The following is a known result about blocks and block-cut point graphs [We96].

Lemma 4.4 *Let $G(V, E)$ be a connected undirected graph.*

(a) Each pair of blocks of G share at most one node, and that node is a cutpoint.

(b) The BC graph of G is a tree in which each leaf node corresponds to a block of G. ■

As before, we focus on obtaining an optimal placement for one service s_j. The role of the BC graph of G in the node resilience design problem is similar to that of BLC graph in the edge resilience design problem. To explain this, we need a few more definitions. A node v of G is an **interior demand point**, if v is a demand point (for service s_j) and v is not a cut point. Note that each interior demand point appears in only one block of G.

Definition 4.6 *Let $G(V, E)$ be a connected undirected graph and let G_B denote the BC graph of G. Consider a service s_j.*

*(a) The **demand subgraph** of G with respect to service s_j, denoted by G_D^j, is the subgraph of G_B consisting of nodes that correspond to blocks of G containing an interior demand point, cut nodes that are also demand points and all edges and nodes of G_B that lie along some path connecting two such nodes.*

*(b) The **pruned demand subgraph** of G with respect to service s_j, denoted by G_{PD}^j, is the subgraph of G_D^j, constructed as follows. If G_D^j consists of a single node, then G_{PD}^j is identical to G_D^j. Otherwise, G_{PD}^j is constructed by removing from G_D^j each leaf node that is a cut point.*

*(c) A demand point v of G is a **generalized interior demand point** if v does not have a corresponding cut point node in G_{PD}^j.*

Since the BC graph G_B of G is a tree (Lemma 4.4), G_D^j and G_{PD}^j are subtrees of G_B. The following lemma shows the relationship between an optimal 1-node-resilient placement for G and the pruned demand graph G_{PD}^j.

Lemma 4.5 *Let $G(V, E)$ be a connected undirected graph and let G_B denote the BC graph of G. Let G^j_{PD} denote the pruned demand subgraph of G with respect to service s_j. Let P^* be an optimal placement for s_j.*

(1) Suppose G^j_{PD} consists of a single node.

(a) If G has only one demand point for s_j, then $|P^*| = 1$.

(b) If G has two or more demand points for s_j, then $|P^*| = 2$.

(2) Suppose G^j_{PD} consists of two or more nodes. Let δ denote the number of leaves of G^j_{PD}. Then, $|P^*| = \delta$.

Proof sketch: Below, we will indicate how a placement P is constructed. The proof that the placement is optimal and that it provides 1-node-resilience can be given in a manner similar to the proofs of Lemmas 4.2 and 4.3. It is omitted in this version due to lack of space.

Suppose G^j_{PD} consists of a single node. There are two possibilities here, as indicated in the statement of the lemma. If G has only one demand point v for service s_j, then P consists of just the node v. If G has two or more demand points, then P consists of two arbitrary nodes from G.

Suppose G^j_{PD} consists of two or more nodes. In this case, we identify the blocks of G corresponding to the leaves of G^j_{PD}. Placement P is obtained by choosing from each such block H, an arbitrary generalized interior demand point. ■

An algorithm for finding an optimal placement for 1-node-resilience can be constructed from the proof sketch given for Lemma 4.5. The steps of the resulting algorithm are shown in Figure 6.

To estimate the running time of the algorithm in Figure 6, let $n = |V|$, $m = |E|$ and let $p = |S|$. The blocks and cut points of a connected graph $G(V, E)$ can be found in $O(|V|+|E|)$ time [CLRS01]. Thus, the BC graph G_B of G can be constructed in $O(n + m)$ time. Consider any service s_j. Using the fact that the pruned demand subgraph G^j_{PD} is a tree, it can be seen that an optimal placement for each service can be found in $O(n)$ time. Hence, the time over all services is $O(pn)$. Therefore, the overall running time of the algorithm is $O(pn + m)$. The following theorem summarizes the above discussion.

Theorem 4.2 *Given a connected graph $G(V, E)$, the set of services S and the set $N(v_i)$ for each $v_i \in V$, the design problem for 1-node-resilience under uniform service costs can be solved in $O(m + np)$ time, where $n = |V|$, $m = |E|$ and $p = |S|$.* ■

5 Concluding Remarks

We identified some resilience metrics for service-oriented networks. These metrics take into account both the underlying topology of the network and the manner in which services are distributed over the network. We presented polynomial algorithms for determining the edge and

Input: A connected graph $G(V, E)$, the set S of all services, the set $N(v_i)$ for each node $v_i \in V$.

Requirement: For each service s_j, find a service point set P_j so that the resulting network is 1-node-resilient and $|P_j|$ is the smallest among all placements that provide 1-node-resilience.

Algorithm:

1. Find the cut points and blocks of G and construct the BC graph G_B.
2. **for** each service $s_j \in S$ **do**
   (a) Compute the demand point set D_j for s_j.
   (b) Construct the pruned demand subgraph G^j_{PD}.
   (c) Construct the service point set P_j by considering the following cases.
   Case 1: G^j_{PD} has only one node.
   If G has only one demand point v for s_j, then let $P_j = \{v\}$. If G has two or more demand points for s_j, choose any nodes x and y of G, and let let $P_j = \{x, y\}$.
   Case 2: G^j_{PD} has two or more nodes.
   (i) Find the leaf nodes of of G^j_{PD} and the corresponding leaf blocks of G.
   (ii) **for** each leaf block H of G **do**
   Choose an arbitrary generalized interior demand point x of H and add x to P_j.
3. Output the sets P_j, $1 \leq j \leq |S|$.

Figure 6: Algorithm for Designing a 1-Node-Resilient Network

node resilience of a given network. We also presented efficient algorithms for optimally distributing services over a given network so that the resulting service-oriented network achieves 1-edge-resilience or 1-node-resilience.

We close by pointing out some directions for future research. First, it is of interest to investigate whether there are asymptotically faster algorithms for determining edge and node resilience. Second, it would be useful to study the design problems for edge and node resilience values larger than one. Third, other versions of design problems (e.g. adding edges of minimum cost to enhance edge or node resilience) can also be studied. Finally, it would also be of interest to identify additional resilience metrics for service-oriented networks and study the corresponding analysis and design problems.

References

[BOS01] G. Brightwell, G. Oriolo and F. Shepherd, "Reserving Resilient Capacity in a Network", *SIAM J. Discrete Mathematics*, Vol. 14, No. 4, Oct. 2001, pp. 524–539.

[CLRS01] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press and McGraw-Hill, Cambridge, MA, 2001.

[CMN03] F. Cuenca-Acuna, R. Martin and T. Nguyen, "Autonomous Replication for High Availability in Unstructured P2P Systems", *Proc. 22nd IEEE Symposium on Reliable Distributed Systems* (SRDS'03), Florence, Italy, Aug. 2003.

[Co87] C. Colbourn, "Network Resilience", *SIAM J. Algebraic and Discrete Methods*, Vol. 8, 1987, pp. 404–409.

[CZ+99] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph and R. Katz, "An Architecture for a Secure Service Discovery Service", *Proc. ACM Symposium on Mobile Computing and Communications* (MobiCom'99), New York, NY, Aug. 1999, pp. 24–35.

[DME02a] C. Dabrowski, K. Mills and J. Elder, "Understanding Consistency Maintenance in Service Discovery Architectures in Response to Message Loss", *Proc. 4th International Workshop on Active Middleware Services* (WAMS'02), July 2002, pp. 51–60.

[DME02b] C. Dabrowski, K. Mills and J. Elder, "Understanding Consistency Maintenance in Service Discovery Architectures During Communication Failure", *Proc. 3rd International Workshop on Software Performance* (WOSP'02), July 2002, pp. 168–178.

[DMR03] C. Dabrowski, K. Mills and A. Rukhin, "Performance of Service-Discovery Architectures in Response to Node Failures", *Proc. 2003 International Conference on Software Engineering Research and Practice* (SERP'03), June 2003, pp. 95–101.

[DW01] J. Douceur and R. Wattenhofer, "Optimizing File Availability in a Secure Serverless Distributed File System", *Proc. 20th IEEE Symposium on Reliable Distributed Systems* (SRDS'01), New Orleans, LA, Oct. 2001, pp. 4–13.

[Ev79] S. Even, *Graph Algorithms*, Computer Science Press, Rockville, MD, 1979.

[GBI03] S. Goel, S. Belardo and L. Iwan, "A Resilient Network that can Operate Under Duress: Supporting Communication Between Government Agencies During Crisis Situations", *Proc. Hawaii Intl. Conference on System Sciences*, Jan. 2003.

[GJ99] P. Georgatsos and Y. Joens (Editors), "Towards Resilient Networks and Services", ACTS Guidelines NIG-G5, June 1999.

[GL03] B. Gedik and L. Liu, "Reliable Peer-to-Peer Information Monitoring Through Replication", *Proc. 22nd IEEE Symposium on Reliable Distributed Systems* (SRDS'03), Florence, Italy, Aug. 2003.

[GTS03] S. Goel, S. Talya and M. Sobolewski, "Service-Based P2P Overlay Network for Collaborative Problem Solving", submitted for publication, March 2003.

[He99] B. Helvik, "Dependability Issues in Smart Networks", *Proc. 5th IFIP Conference on Intelligence in Networks*, Thailand, Nov. 1999, pp. 53–76.

[HH93] F. Harary and J. Hayes, "Edge Fault Tolerance in Graphs", *Networks*, Vol. 23, No. 2, March 1993, pp. 135–142.

[Hw94] F. Hwang, "Comments on 'Network Resilience: A Measure of Network Fault Tolerance'", *IEEE Trans. Computers*, Vol. 43, No. 12, Dec. 1994, pp. 1451–1452.

[IF01] A. Iamnitchi and I. Foster, "On Fully Decentralized Resource Discovery in Grid Environments", *Proc. Intl. Workshop on Grid Computing*, Denver, CO, Nov. 2001.

[Ja94] P. Jalote, *Fault Tolerance in Distributed Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[JW+00] S. Jha, J. Wing, R. Linger and T. Longstaff, "Survivability Analysis of Network Specifications", *Proc. 2000 IEEE International Conference on Dependable Systems and Networks* (DSN'00), Workshop on Dependability Despite Malicious Faults, New York, NY, June 2000.

[LK+03] D. Loguinov, A. Kumar, V. Rai and S. Ganesh, "Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience", *Proc. ACM SIGCOMM'03*, Karlsruhe, Germany, Aug. 2003, pp. 395–406.

[MKG03] L. Massoulie, A. Kermarrec and A. Ganesh, "Network Awareness and Failure Resilience in Self-Organizing Overlay Networks", *Proc. 22nd IEEE Symposium on Reliable Distributed Systems* (SRDS'03), Florence, Italy, Aug. 2003.

[MOY97] S. Moitra, E. Oki and N. Yamanaka, "Some New Survivability Measures for Network Analysis and Design", *IEICE Trans. Communications*, Vol. E80-B, No. 4, Apr. 1997, pp. 625–631.

[NG90] W. Najjar and J. Gaudiot, "Network Resilience: A Measure of Network Fault Tolerance", *IEEE Trans. Computers*, Vol. 39, No. 2, Feb. 1990, pp. 174–181.

[Pr86] D. Pradhan (Editor), *Fault-Tolerant Computing: Theory and Techniques*, Volumes I and II, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[Si01] M. Singhal, "Research in High-Confidence Distributed Information Systems", *Proc. 20th IEEE Symposium on Reliable Distributed Systems* (SRDS'01), New Orleans, LA, Oct. 2001, pp. 76–77.

[SP03] U. Saif and J. Paluska, "Service-Oriented Network Sockets", Technical Report, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[SW97] M. Stoer and F. Wagner, "A Simple Min-Cut Algorithm", *J. ACM*, Vol. 44, No. 4, July 1997, pp. 585–591.

[We96] D. West, *Introduction to Graph Theory*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1996.