ALGORITHMS FOR ANALYZING OF THE RESILIENCE IN COMMUNICATION NETWORKS AND P2P OVERLAY

Dorina Luminiţa COPACI, Constantin Alin COPACI

lcopaci@yahoo.com, acopaci@yahoo.com

Abstract

In this paper, we present a survey of strategies to improve resilience in communication networks as well as in P2P overlay networks. By using some basic concepts from graph theory, we show that many concepts for communication networks are based on well-known graph-theoretical problems. P2P overlay networks evidently benefit from resilience-enhancing strategies in the underlying communication infrastructure, but beyond that, their specific properties pose the need for more sophisticated mechanisms.

Keywords: Peer-to-peer overlay, edge resilience, node resilience

1. INTRODUCTION

The network resilience - the ability to provide and maintain an acceptable service level in the presence of (random or deliberate) failures - becomes more and more important. A resilient network should be able to cope with a specific amount of failures by remaining completely functional, providing connectivity to all of its parts and providing enough capacity to fulfill its task.

Resilience can be achieved either reactively by *restoration* or pro-actively by *protection* methods. Restoration requires a reaction only upon the occurrence of an error. Protection in contrast prepares means of correction through additional redundant information before a failure occurs, and often does not even need retransmissions.

Protection and restoration methods usually apply the following steps:

1. *Failure Detection*
2. *Failure Localization (and Isolation)*
3. *Failure Notification*
4. *Recovery (Protection or Restoration)*
5. *Reversion (Normalization)*

We relate the approaches to known concepts of graph theory, and also try to show interactions, similarities and differences between approaches for communication networks and P2P networks.

This article is organized as follows: in section 2, background on important graph properties and possible graph classes in communication networks is given. In section 3, resilience-enhancing methods for P2P overlay networks are presented and differences to communication networks are identified. Section 4 presents algorithms for analyzing resilience of a given network. Finally, section 5 summarizes the main findings of the article.

2. GENERAL NOTIONS ABOUT GRAPH THEORY

Any network can be modeled as a (directed) graph G consisting of vertices or nodes V and edges or links E. Edges may be weighted to either represent communication capacities, or communication costs or delays.

Resilience was defined as the ability to maintain a network service under interference. Since many of these services depend on the reach ability of nodes, connectivity measures certainly belong to the most important graph properties.

The *edge connectivity* λ and the *vertex connectivity* κ are the minimum number of edges (vertices) that need to fail, to separate the graph into at least 2 components and hence are worst-case statistics of resilience. So, $\lambda - 1$ and $\kappa - 1$ are the numbers of edges (vertices) which may always be removed, without disconnecting the graph. The edge connectivity equals the size of a minimum cut of the graph and is bounded from above by the minimum degree of a vertex.

A graph is called k*-edge-connected* if $\lambda \geq k$, i.e. between every pair of vertices exist at least k edge-disjoint paths. It is called k*-vertex-connected* if $\kappa \geq k$, i.e. between every pair of unconnected vertices there are at least k vertex-disjoint paths.

Another connectivity related measure is the *fragmentation* of a graph. Since, very often the disconnection of a single weakly connected vertex is of minor importance for the whole network, the fragmentation determines a value pair describing the size and relation of its disconnected components. Let $s_1, s_2, .. s_c$ be the number of vertices in the c components of the graph, then the value $frag_1 := \frac{\max_{i=1}^{c} s_i}{\sum_{i=1}^{c} s_i}$ is the relative size of the largest component and the value $frag_2 := \frac{\sum_{i=1}^{c} s_i - \max_{i=1}^{c} s_i}{c-1}$ represents the average size of the remaining components. If the network services are dependent on short communication paths, especially if delays play a role, a second set of statistics, besides the connectivity metrics, becomes important.

The shortest path between two vertices s and t is a set of edges connecting s and t and having a minimum sum of edge weights. Let the distance $d(s,t)$ be the weight of the shortest s-t-path and the distance between unconnected vertices defined to be infinite. The *diameter* of a graph $diam(G) := \max_{s,t \in V} d(s,t)$ then is the length of the longest shortest path between any two vertices. Clearly, the diameter influences the time of information distribution in the whole network.

3. RESILIENCE IN P2P OVERLAY NETWORKS

Peer-to-Peer is a system architecture that describes a service which is distributed over multiple nodes or processes. While in client-server architecture the roles are predefined, all participants generally act as both a client and a server in P2P systems. As these participants usually consist of endhosts, their behavior, arrival, and departure is not well predictable, and possibly very dynamic.

Since an overlay network is a network structure built on top of the communication service of an underlying network, special resilience requirements evolve independently from the resilience of both networks seen alone. To study such effects, it is once more helpful to use the terms of graph theory: An *overlay* of a graph *O = (V,E)* on a communication network *C = (N,L)* is a pair $M = (M_V, M_E)$ consisting of a map $M_V : V \rightarrow N$ of the overlay nodes to nodes of the communication network and a map $M_E : E \rightarrow Paths(C)$, such that $M_E(u,v)$ is a path in *C* from $M_V(u)$ to $M_V(v)$.

A mapping of the overlay, which consists of end-to-end connections onto the underlying communication network, is characterized by its congestion and dilation (sometimes also called *path-stretch):* The *congestion K* of a graph embedding M is defined as the maximum number of overlay paths traversing an edge of the communication network, i.e. $K(M) := \max_{l \in L} \left| \{ e \in E \,/\, l \in M_E(e) \} \right|$. This notion of congestion corresponds to the notion of congestion in communication networks, in the sense, that an edge with a high congestion, tends to be used by a higher number of packets, than edges of low congestion, implying a high risk of communication congestion.

The *dilation D* of an embedding M is the maximum number of edges in any communication path induced by an overlay edge, i.e. $D(M) := \max_{e \in E} \left| M_E(e) \right|$.

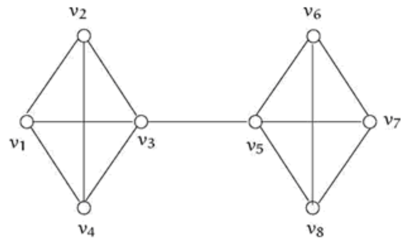
Since, usually, the mapping of overlay nodes to communication network nodes cannot be influenced, the only possibility for optimizations lies in considering the underlying communication topology when constructing overlay edges. This includes the addition of edges in the overlay graph O, allowing alternative routings in the underlying network, which may be used to reduce congestion as well as dilation, and may lead to more efficient communication paths.

The main challenge that arises from the decentralized character of peer-to-peer systems is the distribution of the service to end-hosts rather than to dedicated servers and routers. The highly dynamic character of these end-hosts in comparison to dedicated servers requires peer-to-peer systems to take precautions in order to provide a reliable service. These can be classified by their goals into approaches to

- gain an estimation of the reliability of peers
- provide a reliable routing of requests through
    - redundancy in connectivity, information storage or messaging.
    - imposing a structure on the overlay.

4. ALGORITHMS FOR ANALYZING RESILIENCE OF A GIVEN NETWORK

Consider the eight node network shown in Figure 1.



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	$\{s_5\}$

Figure1. Example to illustrate resilience metrics

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), $1 \leq i \leq 8$. 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 $\{v_1, v_2\}$ $\{v_1, v_3\}$ $\{v_1, v_4\}$, 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, v_3 are deleted, the subnetwork containing only the node v_4 is deficient, since it does not have access to service s_1.

4.1. An Algorithm for Computing Edge Resilience

In this section, we present our algorithm for computing the edge resilience of a given service oriented network.

Input: A network G(V,E), the set S of all services of the network, 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 demand points for service $s_j$;
     c. For each node $v \in D_j$ do
          Compute $\alpha_{v,j}$ , the minimum weight of an s-v edge cutest in $G_j$;
     d. Let $\sigma_j = \min\{\alpha_{v,j} : v \in D_j\}$ .
2. Edge resilience of $G = \min\{\sigma_j : s_j \in S\} - 1$
```

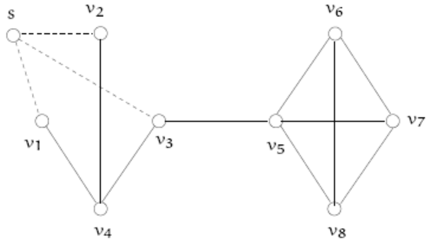


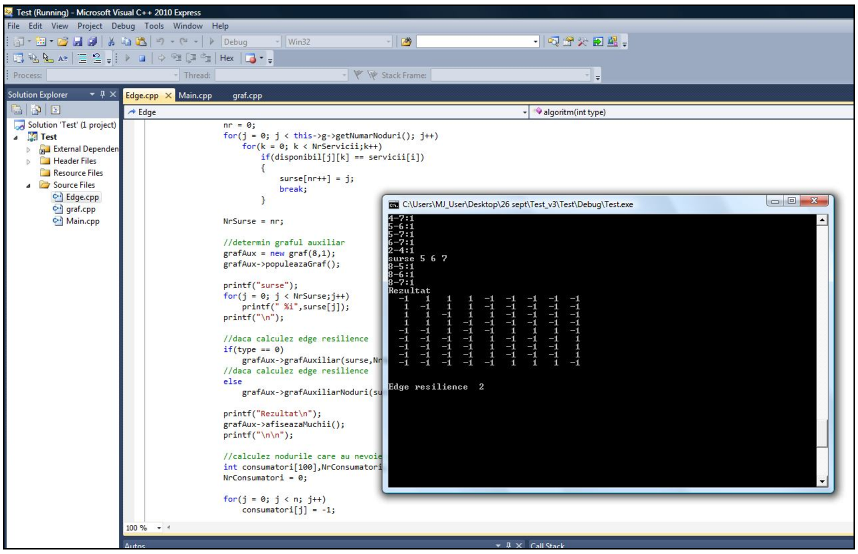
Figure 2. Auxiliary Graph with respect to service s1 of the network in Figure 1

The algorithm for *Computing Edge Resilience* is:

```
Edge::Edge(int servicii[],int NrServicii,int disponibil[100][100],int necesar[100][100])
        {
            g = new graf(8,1);
            g->populeazaGraf();
            g->afiseazaMuchii();
            printf("Initial");
            this->NrServicii = NrServicii;

            int i;
            for(i = 0; i < NrServicii; i++)
                this->servicii[i] = servicii[i];
            int j;
            for(i = 0; i < this->g->getNumarNoduri(); i++)
                for(j = 0; j < this->NrServicii; j++)
                    this->disponibil[i][j] = disponibil[i][j];

            for(i = 0; i < this->g->getNumarNoduri(); i++)
                for(j = 0; j < this->NrServicii; j++)
                    this->necesar[i][j] = necesar[i][j];
        }
void Edge::algoritm(int type)
        {
            //type = 0 -> edge resilience
        ...;
            for(i = 0; i < NrServicii; i++)
            {
                // we determine the source nodes for sj service
                for(j = 0; j < n; j++)
                    surse[j] = -1;

                nr = 0;
                for(j = 0; j < this->g->getNumarNoduri(); j++)
                    for(k = 0; k < NrServicii;k++)
                        if(disponibil[j][k] == servicii[i])
                        {
                            surse[nr++] = j;
                            break;
                        }
        ...;

            //we calculate edge resilience
                if(type == 0)
                    grafAux->grafAuxiliar(surse,NrSurse);
                else
                    grafAux->grafAuxiliarNoduri(surse,NrSurse);

                printf("Matricea de costuri pentru graful auxiliar(ultimul nod corespunde serviciului)\n");
                grafAux->afiseazaMuchii();
                printf("\n\n");
            ....;
                int minAlfa = 100000000,aux;

                // cutset min
                for(j = 0; j < NrConsumatori; j++)
                {
                    type == 0;
                        aux = grafAux->minCutSet(grafAux->getNumarNoduri() -1,consumatori[j],minAlfa,0,0);

                        if(aux < minAlfa)
```

```
                minAlfa = aux;
            }

            if(minAlfa < minSigma)
                minSigma = minAlfa;
        }
        minSigma--;
        printf("Rezilienta muchiei  %i ",minSigma);
    }
```

The result algorithm is:



```
        nr = 0;
        for(j = 0; j < this->g->getNumarNoduri(); j++)
            for(k = 0; k < NrServicii;k++)
                if(disponibil[j][k] == servicii[i])
                {
                    surse[nr++] = j;
                    break;
                }

        NrSurse = nr;

        //determin graful auxiliar
        grafAux = new graf(8,1);
        grafAux->populeazaGraf();

        printf("surse");
        for(j = 0; j < NrSurse;j++)
            printf(" %i",surse[j]);
        printf("\n");

        //daca calculez edge resilience
        if(type == 0)
            grafAux->grafAuxiliar(surse,Nr
        //daca calculez edge resilience
        else
            grafAux->grafAuxiliarNoduri(su

        printf("Rezultat\n");
        grafAux->afiseazaMuchii();
        printf("\n\n");

        //calculez nodurile care au nevoie
        int consumatori[100],NrConsumatori
        NrConsumatori = 0;

        for(j = 0; j < n; j++)
            consumatori[j] = -1;
```

```
4-7:1
5-6:1
5-7:1
6-7:1
2-4:1
surse 5 6 7
8-5:1
8-6:1
8-7:1
Rezultat
  -1   1   1   1  -1  -1  -1  -1  -1
   1  -1   1   1  -1  -1  -1  -1  -1
   1   1  -1   1   1  -1  -1  -1  -1
   1   1   1  -1  -1  -1  -1  -1  -1
  -1  -1   1  -1  -1   1   1   1  -1
  -1  -1  -1  -1   1  -1  -1  -1   1
  -1  -1  -1  -1   1  -1  -1  -1   1
  -1  -1  -1  -1   1  -1  -1  -1   1
  -1  -1  -1  -1  -1   1   1   1  -1


Edge resilience  2
```

4.2. 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.

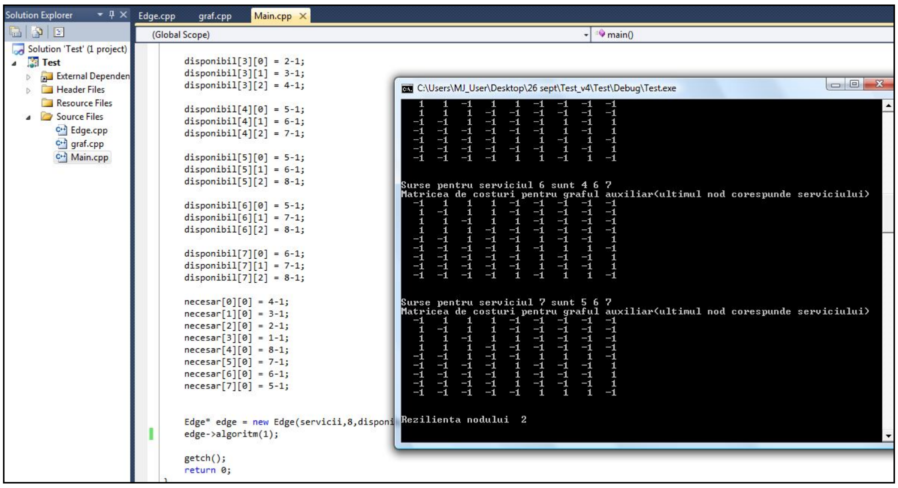
Input: A network G(V,E), the set S of all services of the network, the sets A(v) and N(v) for each node $v \in V$.

Requirement: Find the node resilience of G.

Algorithm:

```
3. for each service $s_j \in S$ do
    a. Construct auxiliary graph $G_j(V_j,E_j)$ 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 cutest in $G_j$;
    d. Let $\Gamma_j = \min\{\gamma_{v,j} : v \in D_j\}$.
4. Edge resilience of $G = \min\{\Gamma_j : s_j \in S\} - 1$
```

The result algorithm is:



```
    disponibil[3][0] = 2-1;
    disponibil[3][1] = 3-1;
    disponibil[3][2] = 4-1;

    disponibil[4][0] = 5-1;
    disponibil[4][1] = 6-1;
    disponibil[4][2] = 7-1;

    disponibil[5][0] = 5-1;
    disponibil[5][1] = 6-1;
    disponibil[5][2] = 8-1;

    disponibil[6][0] = 5-1;
    disponibil[6][1] = 7-1;
    disponibil[6][2] = 8-1;

    disponibil[7][0] = 6-1;
    disponibil[7][1] = 7-1;
    disponibil[7][2] = 8-1;

    necesar[0][0] = 4-1;
    necesar[1][0] = 3-1;
    necesar[2][0] = 2-1;
    necesar[3][0] = 1-1;
    necesar[4][0] = 8-1;
    necesar[5][0] = 7-1;
    necesar[6][0] = 6-1;
    necesar[7][0] = 5-1;


    Edge* edge = new Edge(servicii,8,disponi
    edge->algoritm(1);

    getch();
    return 0;
```

```
   1   1  -1   1   1  -1  -1  -1  -1
   1   1   1  -1  -1  -1  -1  -1  -1
  -1  -1   1  -1  -1  -1   1  -1   1
  -1  -1  -1  -1  -1  -1   1  -1   1
  -1  -1  -1  -1   1   1  -1   1  -1
  -1  -1  -1  -1  -1  -1   1  -1   1
  -1  -1  -1  -1   1   1  -1   1  -1


Surse pentru serviciul 6 sunt 4 6 7
Matricea de costuri pentru graful auxiliar(ultimul nod corespunde serviciului)
  -1   1   1   1  -1  -1  -1  -1  -1
   1  -1   1   1  -1  -1  -1  -1  -1
   1   1  -1   1   1  -1  -1  -1  -1
   1   1   1  -1  -1  -1  -1  -1  -1
  -1  -1   1  -1  -1   1  -1  -1   1
  -1  -1  -1  -1   1  -1   1   1  -1
  -1  -1  -1  -1  -1   1  -1  -1   1
  -1  -1  -1  -1  -1   1  -1  -1   1
  -1  -1  -1  -1   1  -1   1   1  -1


Surse pentru serviciul 7 sunt 5 6 7
Matricea de costuri pentru graful auxiliar(ultimul nod corespunde serviciului)
  -1   1   1   1  -1  -1  -1  -1  -1
   1  -1   1   1  -1  -1  -1  -1  -1
   1   1  -1   1   1  -1  -1  -1  -1
   1   1   1  -1  -1  -1  -1  -1  -1
  -1  -1   1  -1  -1   1   1   1  -1
  -1  -1  -1  -1   1  -1  -1  -1   1
  -1  -1  -1  -1   1  -1  -1  -1   1
  -1  -1  -1  -1   1  -1  -1  -1   1
  -1  -1  -1  -1  -1   1   1   1  -1


Rezilienta nodului  2
```

5. SUMMARY

In this article, we surveyed various concepts for improving resilience in communication networks as well as resilience-enhancing strategies and measures in P2P overlay networks.

The improvement of the resilience of communication networks gives rise to several algorithmic challenges. Especially the construction of alternative routings for the protection of the communication often requires the solution of hard problems and most times can only be achieved by heuristics or on special topologies. Furthermore, most of the proposed strategies rely on high edge and/or vertex connectivity.

P2P overlay networks, realized at application level, require the transmission services of the underlying communication infrastructure, and therefore both benefit from resilience-enhancing measures in the underlying infrastructure.

We examined the key properties of P2P overlays that influence resilience without an in-depth view of application specific details, since there is wide variety of different applications employing P2P overlay techniques. One key property, required for all P2P overlays, is the ability to cope with a very dynamic membership, which requires some precautions to provide a reliable service. Measures for improved resilience can be classified in reliability estimations of peers and in the provisioning of a reliable routing. The latter one in turn can be established based on redundancy in connectivity, information storage or messaging and by imposing a structure on the overlay.

6. REFERENCES

[1] R. Albert, H. Jeong, and A.-L. Barabasi. Error and attack tolerance of complex networks. *Nature,* 406(6794):378-382, July 2000.

[2] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. Science, 286(5439):509–512, October 1999.

[3] M. Brinkmeier, M. Fisher, S. Grau, G. Schafer, T. Strufe: Methods for Improving Resilience in Communication Networks and P2P Overlays

[4] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. Technical Report MSR-TR-2002-82, Microsoft Research, 2002.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Row-stron, and A. Singh. Splitstream: high-bandwidth multicast in cooperative environments. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles,* pages 298-313, New York, NY, USA, 2003. ACM.

[6] F. Chung and L. Lu. The average distance in a random graph with given expected degree. *Internet Mathematics,* 1(1):91-114, 2002.

[7] T. Cicic, A. F. Hansen, and O. K. Apeland. Redundant trees for fast ip recovery. In *Broadnets 2007.* IEEE, 2007.

[8] G. C. Clark and J. B. Cain. *Error-Correction Coding for Digital Communications.* Perseus Publishing, 1981.

[9] clip2. The gnutella protocol specification v0.4. http://rfc-gnutella.sourceforge.net/, 2002.

[10] B. F. Cooper. An optimal overlay topology for routing peer-to-peer searches. In *LNCS: Middleware 2005,* pages 82 - 101, 2005.

[11] P. Elias, A. Feinstein, and C. Shannon. A note on the maximum flow through a network. *IEEE Transactions on Information Theory,* 2:117-119, December 1956.

[12] G. Ellinas, A. G. Hailemariam, and T. E. Stern. Protection cycles in mesh wdm networks. *Selected Areas in Communications, IEEE Journal on,* 18(10):1924-1937, Oct 2000.

[13] W. Grover. *Mesh-Based Survivable Networks. Options and Strategies for Optical, MPLS, SONET, and ATM Networking.*
2004.

[14] A. F. Hansen, A. Kvalbein, T. Cicic, S. Gjessing, and O. Lysne. Resilient routing layers for recovery in packet networks. *International Conference on Dependable Systems and Networks DSN 2005. Proceedings.,* pages 238-247, June-1 July 2005.

[15] T. Klingberg and R. Manfredi. The gnutella protocol specification v0.6. http://rfc-gnutella.sourceforge.net/, 2002.

[16] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Fast recovery from link failures using resilient routing layers. *10th IEEE Symposium on Computers and Communications, ISCC 2005. Proceedings.,* pages 554-560, June 2005.

[17] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Fast ip network recovery using multiple routing configurations. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications,* pages 1-11, April 2006.

[18] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah. Proactive vs reactive approaches to failure resilient routing. *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies,* 1:-186, March 2004.

[19] S. P. M. Shand, S. Bryant. Draft: Ip fast reroute using not-via addresses. February 2008.

[20] M. Medard, S. G. Finn, R. A. Barry, and R. G. Gallager. Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs. *IEEE/ACM Transactions on Networking,* 7(5):641-652, Oct 1999.

[21] J. M. Michael Menth, Andreas Reifert. Self-protecting multi-paths - a simple and resource-efficient protection switching mechanism for mpls networks. *3rd IFIP-TC6 Networking Conference (Networking2004 Athens/Greece),* 2004.

[22] J. Moy. Ospf version 2, apr 1998.

[23] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *ACM Symposium on Parallel Algorithms and Architectures,* pages 311-320, 1997.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications,* pages 161-172, 2001.

[25] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture, jan 2001.

[26] D. J. Rosenkrantz,S. Goel, S. S. Ravi, J. Gangolly: Structure-Based Resilience Metrics for Service-Oriented Networks, October 11, 2004.

[27] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms,* pages 329 - 350, November 2001.

[28] M. Shand and S. Bryand. Draft: Ip fast reroute framework. Technical report, February 2008.

[29] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrish-nan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Aplications. In *ACM Applications, Technologies, Architectures, and Protocols for Computer Communication,* pages 149 - 160, September 2001.

[30] W. W. Terpstra, J. Kangasharju, C. Leng, and A. P. Buchmann. Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search. In *SIGCOMM Comput. Commun. Rev.,* 2007.

[31] S. Wang, D. Xuan, and W. Zhao. Analyzing and enhancing the resilience of structured peer-to-peer systems. *Journal of Parallel and Distributed Computing,* 65:207-219, 2005.

[32] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.