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 article, we present a study of strategies to improve resilience in P2P overlay networks and in communication networks. A lot of concepts for communication networks are based on known graph theoretical problems. P2P overlay networks benefit from resilience increasing strategies in the underlying communication infrastructure.

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

1. INTRODUCTION

The network resilience [3] is the ability to provide and maintain an acceptable service level in the presence of failures. This becomes more and more important. Resilience can be realized 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: f*ailure detection; failure localization; failure notification; recovery (protection or restoration); reversion (normalization)* [3].

We refer the approaches to known concepts of graph theory, and we try to show similitude and differences between approaches for P2P networks and communication network.

This article is organized as follows: In section 2 are given general notions about graph theory. In section 3 are presented resilience improving methods for P2P overlay networks and 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

A network can be modeled as a 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 [3].

The *edge connectivity* λ and the *vertex connectivity* κ [3] 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 $\lambda \geq k$ [3], i.e. between every pair of vertices exist at least k edge-disjoint paths. It is called k*-vertex-connected* if $\kappa \geq k$ [3], i.e. between every pair of unconnected vertices there are at least k vertex-disjoint paths.

The *fragmentation* [3] of a graph is another connectivity measure. The fragmentation determines a value pair describing the size and relation of its disconnected 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].

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 $M_V(v)$.

4. ALGORITHMS FOR ANALYZING RESILIENCE OF A GIVEN NETWORK

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.

We consider a network G(V,E), the set S of all services of the network, the sets A(v), the set of services available at v and N(v), the set of services needed at v, for each node $v \in V$.

We want to calculate the edge resilience of G:

For each service $s_j \in S$ we construct auxiliary graph $G_j (V_j, E_j)$ for service s_j, then find the set of demand points for service s_j. For each node we compute the minimum weight of an *s-v* edge cutest in G_j. Then we compute edge resilience of G.

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 presented in Figure 4.1:

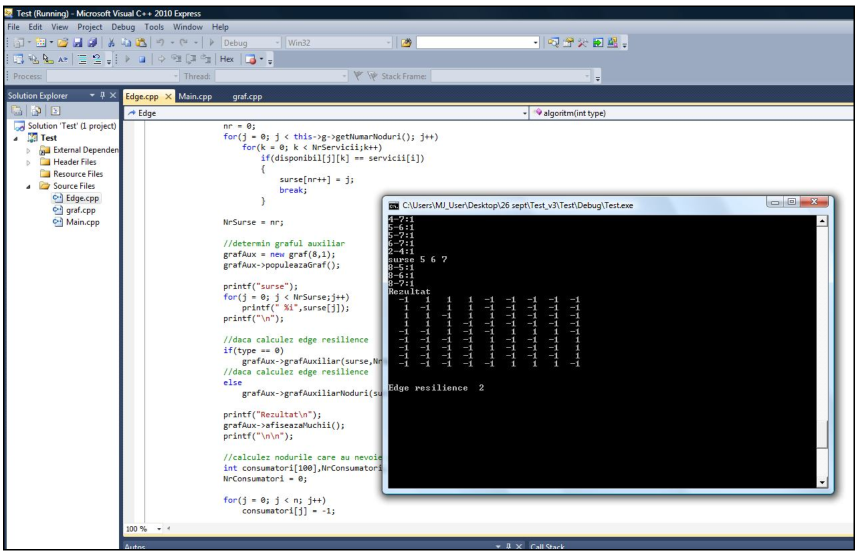


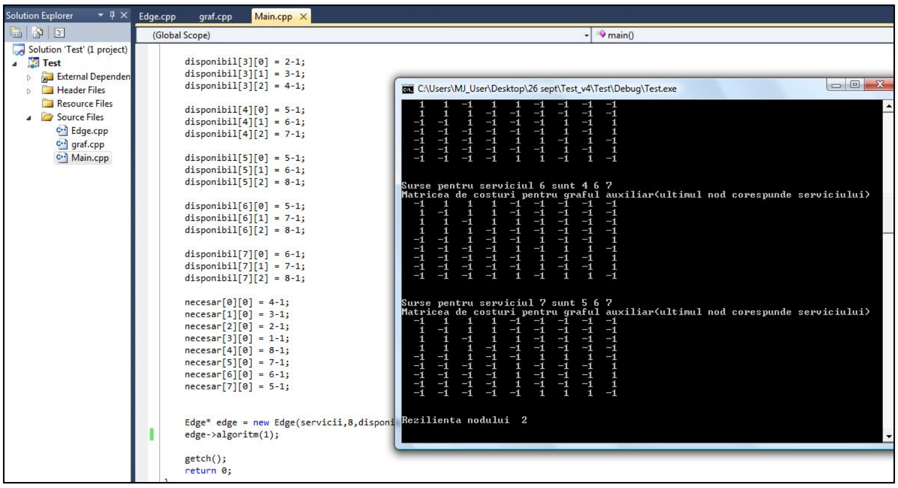
Figure 4.1. The result algorithm for Computing Edge Resilience

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.

We consider a network G(V,E), the set S of all services of the network, the sets A(v) and N(v) for each node $v \in V$. We find the node resilience of G: for each service $s_j \in S$ we construct auxiliary graph $G_j(V_j,E_j)$ for service s_j, then we compute the set of demand points for service s_j. For each node $v \in D_j$ we compute the minimum weight of an *s-v* node cutest in G_j and we determine node resilience.

The result algorithm is:



4.2. The result algorithm for computing node resilience

5. SUMMARY

In this article, we studied different concepts for increase resilience in communication networks and resilience improving strategies in P2P overlay networks.

We examined the key properties of P2P overlays that influence resilience since there is wide variety of different applications employing P2P overlay techniques. One important property for all P2P overlays is the ability to cope with a very dynamic membership, which requires some precautions to provide a reliable service. The reliability estimations of peers and the provisioning of a reliable routing are measures for improved resilience.

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.