Ensuring the Security of a Communication Network through Resilience. Mathematical Modeling

Constantin-Alin COPACI[1], Dorina-Luminiţa COPACI[2]
[1] IT Expert, ANCOM, Bucharest, Romania
acopaci@yahoo.com
[2] Associate Professor, University Politehnica Bucharest - ETTI, Bucharest, Romania
lcopaci@yahoo.com

Abstract

Many of the network computing systems used in various organizations are not resilient enough to withstand attacks and failures. The performance of these networks is degraded by failures. Thus, it is important to develop techniques for designing and implementing resilient service-oriented networks that can survive attacks and failures, as well as continue to provide a reasonable level of service. This paper considers the mathematical modeling using graph theory of resilience in service-oriented communication networks. The objective of this paper is to develop the concept of service-oriented resilient system as well as to identify the metrics used to quantify resilience to node and edge failures. Using these metrics, we will choose an appropriate network topology and/or an optimal distribution of services in the network.

Index terms: edge resilience, graph, node resilience, restoration, service-oriented network

1. Introduction

The resilience [1], [2] approach can be compared to ensuring the quality of service. Network operators can provide resiliency as an added value to the service. In addition, important, critical, driving and emergency usable services can be protected with an additional layer of protection against disasters or terrorist attacks.

Further on, we will consider resilient systems as systems that restore their function after a failure. Thus, such systems have an intrinsically high dependability. The term dependability is a complex notion that includes the following topics: reliability, availability, confidentiality, safety and security [3]. One of the most widely used techniques for increasing the dependability of systems is the implementation of the fault-tolerance [4], [5] a technique to be considered in this paper.

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 number 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 proactively by ***protection*** methods [6]. 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.

Service-oriented communication networks, which we will focus on next, have interesting properties in terms of resilience [7]. This research provides a decomposition of the resilience concept

into two categories: challenge tolerance and trustworthiness. This is graphically illustrated in the figure 1 [8]:

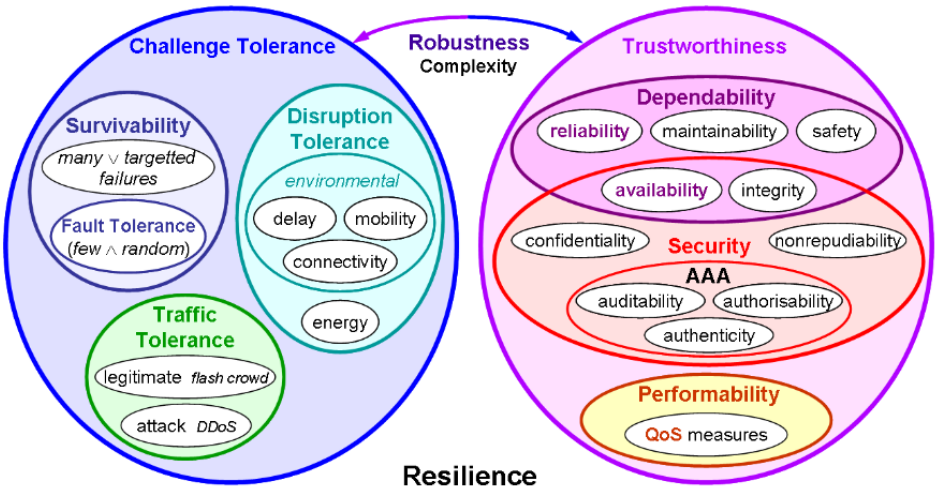

Fig. 1. Challenge tolerance and trustworthiness [8]

This article is organized as follows: In Section 2, information is presented on important graph properties and possible classes of graphs in communication networks. In Section 3, methods to improve resilience for service-oriented communication networks are presented. In this section, we also present algorithms for analyzing the resilience of a service-oriented network. Finally, Section 4 summarizes the main findings of the article.

2. Graph theoretical background

Any network can be modeled as a (directed) graph G consisting of vertices or nodes V and edges or links U. Edges may be weighted to either represent communication capacities, or communication costs or delays [9], [10].

Resilience is 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 α and the vertex connectivity μ 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, α-1 and μ-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.

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 $diam(G) = max_{s,t\epsilon V} d(s,t)$ 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. The resilience in service-oriented networks

A service-oriented architecture [11], [12], [13] involves breaking an application's functionality into smaller, distinct units - called services - that can be distributed across a network and used together to create business applications. The high capacity with which these services can be reused in different applications is a characteristic of service-based architectures. These services communicate with each

other by sending information from one service to another. In such an architecture, each computing system is associated with two sets of services, local services and network services.

3.1. Formal models of the formulated problems

In this part of the research, we developed formal models of the formulated problems. On this basis, we implemented a series of algorithms for the design of resilient edges, respectively of resilient nodes.

We consider the graph G (V, U) that abstracts the network. This is used to design a resilient network. The set V represents the set of nodes of the graph corresponding to the network, and U the set of edges in the graph G.

For each node $v_i \in V$ and for each edge $u_{ij} \in U$ we define the set $N(v_i)$ of services required by each v_i. We denote the cost matrix by $C=[c_{ij}]$, where c_{ij} represents the cost of introducing service s_j at node v_i. The cost matrix is a matrix with positive elements.

$$c_{ij} = \begin{cases} 0, & \text{if there is no service introduction cost} \\ 1, & \text{if there is a service introduction cost} \end{cases}$$

We determine the set of services for each node in the graph so that the resulting network has the required level of edge or node resilience and the total cost of placing the services is minimal. In this sense, we consider s_j a certain service. We define each node v_i such that $s_j \in V(v_i)$ is a demand point for s_j. The node for which the service s_j exists is called a *service point*. A lot of service points for s_j is called *placement* for s_j.

Given the connected graph G (V, U) and a placement P for the service s_j, then the placement will be the edge resilience with respect to the service s_j if for each edge u ∈U, the graph G'(V,U-{u}) contains a road from each demand point for s_j to a service point for s_j.

We consider the ten nodes network shown in Figure 2.

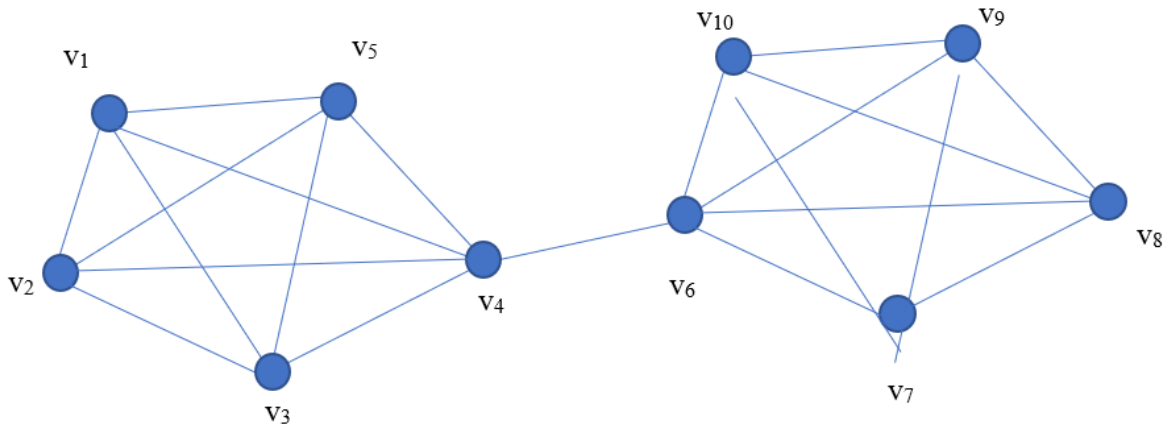

Fig. 2. The graph representing a network with 10 nodes

The ten services provided by the network are denoted by s_1 through s_{10}. For each node v_i, we define the set of services available to v_i respectively the set of services required by the node v_i, $1 \le i \le 10$. For example, for i=1, the set of services available to v_1 is $\{s_1,s_2,s_3,s_4\}$ and the set of services required by the node v_1 is $\{s_5\}$. 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_4) or one edge (the edge $\{v_4,v_6\}$). However, the node and edge resilience parameters of the network are both two. In particular, the subnetworks obtained by deleting the edge $\{v_4,v_6\}$ or one of the nodes v_4 and v_6 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 four edges $\{v_1, v_2\}$, $\{v_1, v_3\}$, $\{v_1, v_4\}$, $\{v_1, v_5\}$, 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, v_4 are deleted, the subnetwork containing only the node v_5 is deficient, since it does not have access to service s_1.

3.1.1. Mathematical modeling for the resilience of network edges

In this section, we implement the algorithm for calculating the edge resilience of a service-oriented communication network. Also starting from the definitions in graph theory, we considered G (V; E) the graph of the given network. We used the term subnet to understand a connected subgraph of G.

A service-oriented network subnet is defective with respect to a network service if there is a node in the subnet that requires the given service, but there is no node in the subnet that provides that service. In this situation, a subnet is deficient if there is a service for which the subnet is deficient.

Determining the resilience of service-oriented communication network edge presupposes the determination of the set of available services as well as the set of services required for each node [14], [15].

The minimum number of deficient edges related to the service s_i can be obtained by calculating the minimum cut of the edge with the minimum weight in the auxiliary graph G_i for each pair s-v, where v is a node that requires the service si. Once this value is found, the edge resilience of the given network G can be determined by considering the minimum for all services. These observations lead to the algorithm [14], [15] for computing the edge resilience of a given service-oriented network.

To implement the algorithm we will consider a network G(V, E), the set S of all services of the network, the set of available services as well as the set of services required for each node $v \epsilon V$. We will find the edge resilience of G.

The algorithm assumes that for each service $s_i \in S$ to construct auxiliary graph G_i (V_i, E_i) for service s_i; to find the set $D_i \in V_i$ of demand points for service s_i;

- For each node $v \in D_i$ to compute $\alpha_{v,i}$, the minimum weight of an s-v edge cutest in G_i;
- Let. $\sigma_i = min\{\alpha_{v,i} : v \in D_i\}$. Edge resilience of $G = min\{\sigma_i : s_i \in S\} - 1$.

Determining the edge resilience of a graph that abstracts a service-oriented communication network:

```
[…]
Edge::Edge(int servicii[],int NrServicii,int disponibil[100][100],int necesar[100][100])
                {       g = new graf(10,1);
                        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);}
```

To estimate the running time of the algorithm, we assumed that the given network has x nodes, y edges and a total of r services. We determined the running time of the algorithm by calculating the minimum cut of the edge with minimum weight. For each service si the algorithm computes the minimum cut O(x). Thus, the total number of calculations is O(rx). Since each computation of the minimum edge cut can be done in O(y+xlogx) time, the running time of the algorithm is O(rx(y+xlogx)).

3.1.2. Mathematical modeling for the resilience of network nodes

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.

As in the case of determining the resilience of the edge, and in the case of calculating the resilience of the node, we used the auxiliary graph, except that the weights of the crowd were not used, and the weight of each node was considered equal to 1.

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 set to obtain the definition of a deficiency inducing node set.

We considered a service-oriented network G (V, E) and sj a given service. We denote by Gj(Vj,Ej) the auxiliary graph (with node weights) for sj. For any node, the minimum number of nodes to be deleted from G such that there is no path between v and any node providing service sj is equal to the weight of the minimum cut s-v with the minimum weight in Gj [14], [15]. The rest of the calculation is similar to that of edge strength.

4. Summary

This article provides an overview of resilience mechanisms in service-oriented communication networks. We reviewed different concepts for improving resilience in service-oriented communication networks. We developed the concept of a service-oriented system resilient to node and edge failures and determined the metrics for ensuring the security and resilience of service-oriented systems.

Different resilience mechanisms have advantages and disadvantages. The benefit of the resilience strategy depends on the specifics of the network and how much importance a network operator attaches to performance metrics.

References

[1]. V. Kuikka, Modeling Network Resilience and Utility of Services. In Proceedings of the 2019 IEEE International Systems Conference (SysCon), Orlando, FL, USA, 8–11 April 2019.
[2]. http://en.wikipedia.org/wiki/Resilience.
[3]. I.C. Bacivarov, V. Cătuneanu, Fiabilitatea sistemelor de telecomunicaţii, Ed. Militară, 1995.
[4]. xxx Proceedings of IEEE Symposium on Fault Tolerant Computers, 1990-2010.
[5]. R. Albert, H. Jeong, and A.-L. Barabasi, Error and attack tolerance of complex networks, *Nature*, 406(6794):378-382, July 2000.
[6]. 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*, March 2004.
[7]. D. J. Rosenkrantz,S. Goel, S. S. Ravi, J. Gangolly: Structure-Based Resilience Metrics for Service-Oriented Networks, October 11, 2004.
[8]. https://www.enisa.europa.eu, “Measurement Frameworks and Metrics for Resilient Networks and Services: Technical report”, February 2011.
[9]. M. A. Henning, J. H. van Vuuren, Graph and Network Theory – An Applied Approach using Mathematica, Springer Cham, ISBN: 978-3-031-03857-0.
[10]. F. Chung and L. Lu. “The average distance in a random graph with given expected degree.” Internet Mathematics, 1(1):91-114, 2002.
[11]. M. Aly; M. Franke (2016). "Service Oriented Interactive Media (SOIM) Engines Enabled by Optimized Resource Sharing". 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE): 231–237. Retrieved February 9, 2021.
[12]. F. Glinka; A Raed (2009). "A Service-Oriented Interface for Highly Interactive Distributed Applications". European Conference on Parallel Processing. ISBN 978-3-642-14121-8. Retrieved February 9, 2021.
[13]. Dieter Hildebrandt; Jan Klimke (2011). "Service-oriented interactive 3D visualization of massive 3D city models on thin clients". COM.Geo '11: Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications. COM.Geo '11: 1. doi:10.1145/1999320.1999326. ISBN 9781450306812. S2CID 53246415. Retrieved February 9, 2021.
[14]. D. J. Rosenkrantz, S. Goel S. S. Ravi, J. Gangolly, “Structure-Based Resilience Metrics for Service-Oriented Networks”, October 2004.
[15]. 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,* June 2005.