Network Partitioning and Self-sizing Methods for QoS Management with Autonomic Characteristics

Romildo Martins da Silva Bezerra[1] and Joberto Sérgio Barbosa Martins[2]

[1] Federal Institute of Education, Science and Technology of Bahia (IFBA)
Salvador – Bahia – Brazil
romildo@ifba.edu.br
[2] Salvador University (UNIFACS)
Salvador – Bahia – Brazil
joberto@unifacs.br

Abstract. The increasing complexity, heterogeneity and unpredictability of networks make the task of managing these systems highly complex. The autonomic computing paradigm brings innovative solutions to the network management area with new adaptable and "clever" solutions which should consider a heterogeneous functionality in a wide variety of fields. One of the challenges involved in proposing autonomic solutions for QoS management consists of dealing with the inherent complexity and proposing solutions with feasible execution time. In brief, the autonomic solution effective response time should be fast enough so that it could be applied before any new important network state change. In this paper, a framework that uses self-partitioning methods is considered as the basis for evaluating a solution to the problem of dynamic LSPs setup allocation in a general MPLS network topology focusing on keeping a low complexity solution and aiming at achieving the best possible problem solution identification response time.

Keywords: Network Management, Network Partitioning, Self-sizing Networks, Bandwidth Allocation, Autonomic Computing, Computational Complexity, Quality of Service, MPLS, LSPs.

1 Arguing on the Basic Characteristics of Autonomic Frameworks - An Introduction

The identification of an autonomic computing solution typically requires the definition of an autonomic framework which considers the autonomic paradigm main requirements such as adaptability, management of heterogeneous functionality and to promote intelligent interactions by using learning and reasoning techniques [1]. Besides that, one of the major challenges existing when proposing an autonomic solution is that they typically have to deal with highly complex networks and should have a feasible execution time. Indeed, the needed autonomic approach has to be computed in a feasible time in order to generate a "possibly

C.S. Hong et al. (Eds.): APNOMS 2009, LNCS 5787, pp. 151–160, 2009.

valid solutions" applicable to the network being managed. Thus, we have to look for a possible set of solutions and methods arranged in an adequate autonomic framework such that we try to reduce the inherent complexity and, at same time, we try to improve execution time performance in order to get a feasible execution time. As such, approaches applied to lower network complexity and to promote feasible execution time or improved performance are considered as "autonomic characteristics" and are recommended for an autonomic framework.

Network self-sizing capabilities have been explored in global and local approaches [2]. In brief, self-sizing consists in employing the ability to compute and allocate network resources using either a global or a local perspective. Discovering community structures or network clusters in huge and complex networks has been extensively studied by the research community [3] [4] [5] [6] . Network clustering algorithms have been proposed with great success in social networks, collaboration networks, gene networks, among others.. As a result, algorithms which discover communities in networks are able to partitioning the network and could be adjusted and/or adapted to take into account the autonomic system's complexity and response time requirements existing for autonomic network management.

In brief, the approach proposed by this paper consists of applying the self-sizing and partitioning methods considering the scope of an autonomic network management framework in order to achieve a less complex computational scenario resulting in a feasible execution time for the autonomic solution. The fact of being able to compute in an autonomic way in a complex network topology scenario is considered a basic characteristic that autonomic systems should be engaged in. The paper evaluates the proposed partitioning and self-sizing approaches by establishing a huge amount of LSPs in a MPLS network with a significant number of nodes and random topology.

In the next sections the autonomic network management framework is introduced. In section 3 we argue on the use of self-sizing and partitioning methods in autonomic frameworks. Section 4 evaluates a group of partitioning algorithms and proposes a new partitioning algorithm which considers the autonomic characteristics required. Section 5 evaluates the partitioning algorithm proposed considering the establishment of a set of LSPs in a random MPLS network with a significant number of nodes. Section 6 presents the final considerations and future works.

2 Autonomic Network Management Framework

The basic framework model adopted [7] to support dynamic resource allocation algorithms with autonomic capabilities is able to manage quality of service in computer networks using a Full Policy-based Management (FPBM) approach as shown in Figure 1. The model uses a traditional Policy-based Management strategy [8] to manage the policy lifecycle. To create new policies in an autonomic way according to the new state of the network it is added a knowledge layer, replacing the policy editor typically used in Policy-based Management systems.

By adopting this structure, the policies used by the execution plane are generated dynamically by the decision plane, replacing human intervention in this function. This new autonomic management model divides the autonomic network management for quality of service (QoS) in three planes: information, decision and execution planes. The model planes structure and interrelation are show in Figure 1.

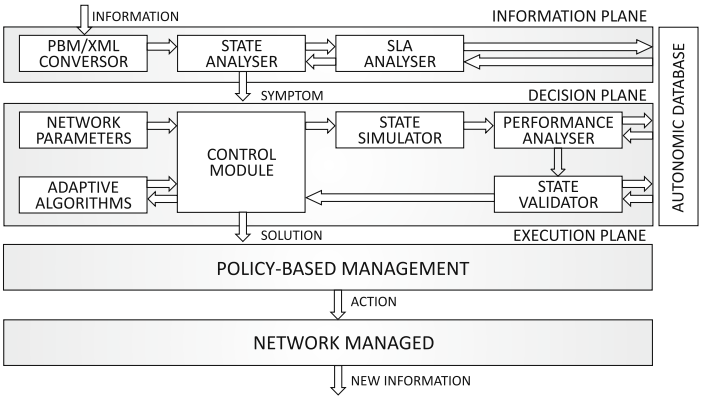


Fig. 1. Autonomic Management Network Model [7]

The self-management process runs in a cyclic way, i.e. the network continuously report its state, independently of the occurrence of problems, thus enabling self-optimization and supporting problems prevention (self-healing). The network state is analyzed (information plane), a solution is found (decision plane), and this solution is converted into a high-level policy (execution plane). The policies are translated into network technologies and mapped according with device capabilities. The policy distribution is done by a Policy Decision Point.

It follows a brief description for each plane.

- Information Plane - The information plane receives data which contains the network state and converts it to a standardized format in a high-level language (XML format) that can be treated by the framework. This conversion provides extensibility to the model since the creation of new converters makes it compatible with other sub-systems or applications. After that, the current network state (snapshot) is analyzed in order to verify its accordance with the Service Level Agreement (SLA) looking for the maintenance of pre-defined network parameters being managed.
- Decision Plane - The decision plane receives the network symptom and finds a solution that meets the specified SLA, if it exists. It is important to observe that the search for a solution, at first, considers an optimum solution which is the best solution for a set of solutions. Meanwhile, the overall algorithm complexity and time limitation in the searching process are considered. For example, an optimum solution to a problem that is no longer occurring is not the best strategy to be used.

– Execution Plane - The execution plane receives the diagnosis (solution) and generates a policy through the policies compiler. Before being implemented, the policy goes through a phase of syntax and semantics validation that does not compromise the autonomic information base.

In terms of autonomic network management model described, the decision plane is the focus. As indicated, the overall algorithm complexity and time limitation in the searching process for the best solution have to be considered. The following sections argue on the use of self-sizing and partitioning methods as a possible solution to deal with these issues.

3 Using Network Self-sizing and Partitioning Capabilities to Support Autonomic Management with Dynamic Resource Allocation

One of the expected functionalities that should be considered in autonomic management system design is its ability to support a dynamic resource allocation approach. Various dynamic resource allocation algorithms have been proposed and, in brief, they provide optimized results based on various criteria and computed based on near real time traffic data collected with various different measurements approaches. When considering an autonomic management framework, the dynamic resource allocation algorithm presents an additional requirement: it should scale to support large and complex network topology structures while keeping an "affordable" execution time. In our novel autonomic network management model, we propose an adaptable solution at the decision plane which promotes the utilization of dynamic resource allocation algorithms by applying simultaneously self-sizing and partitioning principles to the network node structure.

Networks with self-sizing capabilities have the ability to allocate resources like link capacity automatically and adaptively using online gathered data traffic information. Resource allocation decisions can be realized on self-sizing networks "globally" or "locally". A global self-sizing approach means a more centralized solution typically based on a snapshot of the network. A local self-sizing approach means that individual nodes should compute their new states based, typically, on locally acquired information.

In [2], it is argued that an important aspect of the self-sizing capability is the computation time required to find the optimal allocation. This computation time depends on the efficiency of the algorithm, and also on the number of possible solutions for the allocation problem. The number of possible allocations N can be represented as follows:

$$N = k^{lm} \tag{1}$$

where k, l and m are the routing alternatives, services supported and number of source-destination (SD) pairs respectively. As such, the inherent complexity and resulting computation time have a strong dependency on the network node structure and grouping or communities.

Partitioning is a second principle proposed to be used in our autonomic framework in order to better support its autonomic characteristics while making use of dynamic resource allocation algorithms. Partitioning corresponds to the identification of community structures in networks. According to [4], community structure is a property argued to be common in many networks corresponding to the division of network nodes into groups within which the network connections are dense, but between which they are sparse. Community structures in networks have been extensively studied and the concept is closely related to the idea of graph partitioning in graph theory [3] [4] [5] [6]. In terms of the autonomic network management model support for the adaptable use of dynamic resource allocation algorithms the objective is to develop a recursive algorithm that searches a maximum group of k communities with a maximum size of m nodes keeping connections among k groups as sparse as possible. In network terms this means that the partitioning algorithm should as much as possible to minimize network traffic among partitioned communities.

As such, the set of principles adopted for the autonomic framework model is to apply both partitioning and self-sizing methods in order to reduce the inherent complexity and to promote an improved overall response time in complex computer network structures.

4 A Recursive Partitioning Algorithm Based on Network Topology Density

A computer network is an undirected connected graph $G = (V, E)$ of sets such that $E \supseteq |V^2|$, thus, the elements of E are two-element subsets of V[1]. In graph theory, the elements of V are the vertices (nodes or points) of graph G and the elements of E are its edges (lines).

In this paper will use the following notations and definitions:

- The order of a graph G is $|V|$ (vertex number). In this paper $n = |V|$.
- The vertex degree is the number of edges that connect to it.
- The density of a graph is the ratio between the number of edges and the number of possible edges.

In general, the task of finding an exact solution to a graph partition is considered a NP-complete problem, making difficult to solve it for huge network structures. Many areas such as social networks, web graph, ecosystems and others use graphs partitioning algorithms to find common properties or relationships between the vertices. The property that seems to be common to many networks is to define community structures. In other words, the main objective is to divide network vertices into sub-graphs that have higher density.

When looking for higher density sub-graphs, one obtains a set of vertex where the relationship between them is stronger, i.e., the number of edges that one is larger. In the context of computer networks, this represents a greater number

[1] Edges of type (e_i, e_i) will not be considered and (e_i, e_j) is equal to (e_j, e_i).

of paths between a vertices set. These vertices subsets of higher density are seen as sub-domains. The edges that do not belong to any sub-domain are links that allow inter-domain communication. As such, the algorithm originally used for finding communities will be effectively used in our context to identify the vertices for sub-domain creation. The algorithm will receive a graph G (network topology) and number n indicating the required number of sub-domains arrangements and then creates a new graph with the interconnection of communities, in effect, sub-domains (Figure 2).

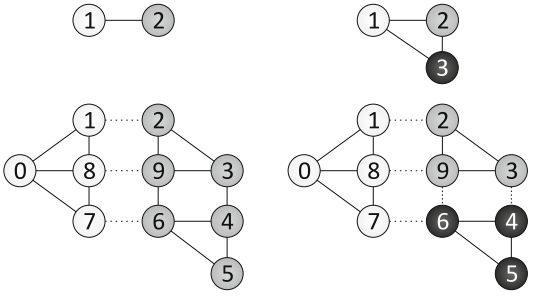


Fig. 2. Graph with two partitions and three partitions

The algorithms "walktrap"[2] [3], "fast greedy" [6], "eigenvector" [5] and "edge betweenness" [4] were initially evaluated to find communities (partitioning nodes) from large computer network domains. The algorithm evaluation was based on two metrics classified in terms of their relevant impact for dynamic resource allocation complexity and execution time algorithms support: minimum number of edges not belonging to the domain (higher density sub-domains) and execution time. The eigenvector algorithm was immediately discarded because it does not generated sub-graphs from certain pre-defined network sizes. The evaluation was then carried out with related graphs, not complete, with the minimum degree equal to two and randomly generated with n equal to 10, 50 and 100. The results are indicated in Table 1.

The "edge betweenness" algorithm showed the best behavior and was chosen as the algorithm to be used as the basic block towards the generation of sub-graphs with high average density, considering the three parameters analyzed (execution time, density and sub-domains size standard derivation). In brief its partitioning into sub-domains resulted in a maximum number of edges within the generated sub-graphs and the algorithm presented a satisfactory execution time. Moreover, the variation in the sub-graphs cardinality was the lowest.

As the next step, it is necessary to have a new algorithm that considers complexity and execution time in order to adequately support autonomic characteristics. A new "edge betweenness with dense sub-domain partitions" algorithm

[2] This algorithm uses random paths and their performance may not remain stable, because it is not deterministic.

Table 1. Algorithms evaluation results considering relevant metrics in order to find communities for huge computer networks. Best choices are in bold.

Parameters	execution time			density			standard deviation		
	seconds			edges not used			sub-domain size		
Algorithms — Nodes	10	50	100	10	50	100	50	100	250
edge betweenness	0.141	**0.172**	0.872	5	**17**	**30**	**2.05**	**6.83**	12.09
fast greedy	**0.125**	0.188	1.106	5	**17**	35	4.27	7.06	14.55
walktrap	0.126	0.181	**0.186**	5	19	33	3.90	11.98	**11.72**

enforcing a greater network density was then implemented in language "R" [9] following the steps illustrated in Algorithm 1.

```
1  input-graph(mynet, k);
   /* group vertices with edge.betweenness in k groups                    */;
2  wtc ← edge.betweenness.community(mynet);
3  mynetcsize ← array(community.to.membership(mynet, wtc$merges,
   steps=find.steps(mynet,k))$csize);
4  mynetmembership ← array(community.to.membership(mynet, wtc$merges,
   steps=find.steps(mynet,k))$membership);
   /* create a list of vectors with subdomains membership                 */
   subdomainslist ← vector("list", dim(mynetcsize));
5  subdomainslist ← create.subdomainslist(mynetcsize,mynetmembership);
   /* create subgraphs                                                    */;
6  subdomain=list();
7  for i = 1 to dim(array(subdomainslist)) do
8      subdomain[[i]] ← subgraph(mynet,array(subdomainslist[[i]]));
   /* subgraphs union                                                     */;
9  subdomainsunion ← graph.empty(n=0, directed=FALSE);
10 subdomainsunion ← subgraphs.union(subdomain, subdomainslist);
   /* identify interdomains edges                                         */;
11 sdiff ← graph.difference(mynet,subdomainsunion);
   /* create the new graph                                                */;
12 mynetup ← newgraph(mynet,subdomainsunion,sdiff);
```

Algorithm 1: The new "edge betweenness with dense sub-domain partitions" algorithm

The proposed "edge betweenness with dense sub-domain partitions" algorithm enforces a "network density" objective. It receives a graph (*mynet*) and a partitions number . It returns to the user a set of high density sub-graphs and creates a new graph (*mynetup*) indicating the relationship between the new set of subdomains (Figure 2).The goal achieved by the derived algorithm is to aggregate the traffic on sub-graphs of higher density, reserving the edges not belonging to any sub-graphs only for communication between domains. Being so, the network is partitioned as the interconnection of a set of high-density subnets.

The "edge betweenness with dense sub-domain partitions" algorithm uses a score for an edge and measures the number of shortest paths through it. This

procedure is repeated for all vertices for . The main idea of this new based community structure detection algorithm is that it is likely that edges connecting separate modules have high edge betweenness as all the shortest paths from one module to another must traverse through them.

5 Self-sizing with Local Control and Partitioning Algorithm Evaluation for LSPs Setup in a MPLS Network

We consider now the allocation of resources using a self-sizing approach at sub-domain level in order to evaluate the effectiveness and performance improvements created by the partitioning approach used. The sequence of actions taken to evaluate the partitioning algorithm used is as follows:

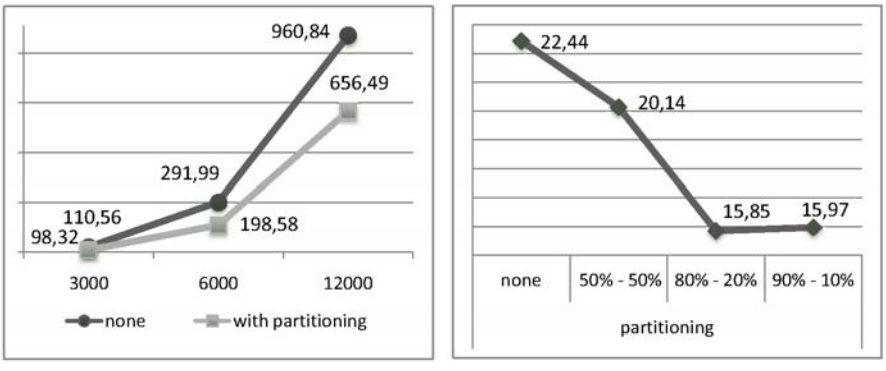
1. The "edge betweenness with dense sub-domain partitions" algorithm receives network data (topology, number of partitions).
2. The "edge betweenness with dense sub-domain partitions" algorithm generates the set of high-density sub-domains (partitioning).
3. A set of sub-graphs is created and interconnection edges are identified (domains interconnections).
4. An allocation resource algorithm is invoked and applied in order to establish a huge set of LSPs through the entire network.
5. The resource allocation algorithm allocates LSPs for intra-domain traffic.
6. The resource allocation algorithm allocates LSPs for inter-domain traffic.
7. The sequence of actions 5 and 6 is repeated whenever the chosen resource allocation algorithm must interact in order to setup LSPs interactively due to restrictions occurring in terms of bandwidth allocation or another allocation parameter constraint.

The resource allocation algorithm used for evaluating the partitioning algorithm was the "penalty-based LSP allocation algorithm" [10]. This algorithm, in brief, allocates LSPs according to their non allocation penalty and required bandwidth assigned by the network manager. In effect, for the specific purpose of evaluating the network partitioning into sub-domains any resource allocation algorithm could be used. Our option was to evaluate the partitioning effect with a dynamic allocation algorithm that will be used in the context of the autonomic network management framework.

The evaluation results are illustrated in Figure 3 considering the following evaluation scenarios. First, the setup of 3.000, 6.000 and 12.000 LSPs respectively. Second, the distribution intra-domains and inter-domains traffic among partitioned sub-domains follows the parameters 50%-50%, 80%-20% and 90%-10% respectively.In the first evaluation scenario (Figure 3a) a set of 3.000 LSPs was established considering the full interconnected network and the partitioned one with different configurations for inter-domain traffic. The execution time evaluated indicated an improvement in the overall execution time achieved with

reductions of approximately 10,25% to 29,37% depending on the inter-domain traffic considered.

In the second evaluation scenario (Figure 3b) sets of 3.000, 6.000 and 12.000 LSPs were setup assuming an inter-domain distribution traffic of 50%-50%. The evaluation results show that the execution time was improved by approximately 11,07% (3.000 LSPs) to 31,68% (with a greater set of established LSPs) by using the partitioning algorithm.



(a) First Evaluation Scenario (b) Second Evaluation Scenario

Fig. 3. LSPs setup allocation execution time evaluation with network partitioning

6 Final Considerations and Future Work

The management of heterogeneous functionality, the need of adaptability and the application of learning and reasoning techniques to support intelligent interaction are considered an essential requirement for achieving autonomic management in most frameworks.

This paper introduces a new partitioning algorithm applicable to the management of quality of service in autonomic computer networks and, by applying the self-sizing method, the evaluation results show that it promotes an overall improvement in the autonomic framework performance (approximately between 10% to 30%) in terms of a specific resource allocation algorithm. The scenario considered was the one where a dynamic allocation resource algorithm is used to set up a significant amount of LSPs in a huge MPLS network generated at random. In effect, dynamic resource allocation is a critical area since autonomic systems should derive on the fly new network states for, typically, highly complex topologies while keeping a feasible execution time to derive these solutions. In other words, the most basic result shown in this paper is that by applying partitioning and self-sizing is possible to improve the autonomic system performance.

Based on the results, authors argue that the partitioning of complex computer network topologies coupled with the capability of self-sizing methods could promote the employment of dynamic resource allocation algorithms to derive new solutions in complex networks and, as such, will pave the road to the creation of more clever and effective autonomic management systems.

In terms of the autonomic framework model proposed, the solution described effectively manages heterogeneous functionality, since the decision layer finds a solution by adopting a set of established methods and paradigms (partitioning and self-sizing) in a specific and focused way. Additional simulations will be carried on considering others dynamic resource allocation algorithms in order to evaluate dependencies with respect to the partitioning algorithm proposed and its performance.

References

1. Jennings, B., van der Meer, S., Balasubramaniam, S., Botvich, D., Foghlu, M., Donnelly, W., Strassner, J.: Towards autonomic management of communications networks. Communications Magazine, IEEE 45(10), 112–121 (2007)
2. Nalatwad, S., Devetsikiotis, M.: Self-sizing networks: local vs. global control. In: Proceedings of IEEE International Conference on Communications, June 2004, vol. 4, pp. 2163–2167 (2004)
3. Pons, P., Latapy, M.: Computing communities in large networks using random walks. Journal of Graph Algorithms and Applications 10(2), 191–218 (2006)
4. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E 69 (2004)
5. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Physical Review E 74 (2006)
6. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E 70 (2004)
7. Bezerra, R.M.S., Martins, J.S.B.: A sense and react plane structured autonomic model suitable for quality of service (qos) management. In: Proceedings of 5th International IEEE Workshop on Management of Ubiquitous Communications and Services, vol. 1 (Maio 2008)
8. Bezerra, R.M.S., Martins, J.S.B.: Functional decoupling principle applied to network device and qos management. In: Proceedings of 7me Colloque Francophone of Gestion of Rseaux et of Services, June 2006, vol. 1, pp. 47–58 (2006)
9. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2009) ISBN 3-900051-07-0
10. Bezerra, R.M.S., Martins, J.S.B.: Penalty-based lsp allocation algorithm. Technical Report TR-01-AN2009, DMCC (2009)