Addressing Stability of Control-Loops in the context of the GANA architecture: Synchronization of Actions and Policies

Nikolay Tcholtchev, Ranganai Chaparadza, Arun Prakash
{Nikolay.Tcholtchev | Ranganai.Chaparadza | Arun.Prakash@fokus.fraunhofer.de}
FOKUS Fraunhofer Institute for Open Communication Systems, Berlin, Germany.

Abstract. As research on self-managing networks is on the rise, a very important question requiring answers is: How to ensure that all autonomic entities/elements in a network or in a single node work together harmoniously with the aim of maintaining service availability and good levels of quality of service (QoS) provided to the end users? In this paper, we propose to enhance the *Generic Autonomic Network Architecture* (GANA) – a recently emerged architectural Reference-Model for Autonomic Networking, such that actions, policy enforcements and/or (re-) configurations, issued by different autonomic (decision making) entities, are synchronized in such a way that they lead to the best possible (long-term) reaction of the system to the challenging conditions the network is exposed to.

Keywords: autonomic behaviors of a control-loop, stability of a control-loop(s), self-managing networks, GANA

1. Introduction

Autonomicity – realized through control-loop structures is an enabler for advanced and enriched self-manageability of network devices and networks. As research on self-managing networks is on the rise, a very important question requiring answers is: How to ensure that all autonomic entities/elements in a network or a single node, work together harmoniously with the aim of maintaining service availability and good levels of quality of service (QoS) provided to the end users? Autonomic entities/elements normally control only some of the diverse aspects that are significant for self-manageability of networks. In a complex environment or architecture involving several control-loops that need to execute in parallel, the challenge is to ensure that the autonomic elements' behaviors are synchronized towards a global network goal. That is required in order to avoid a situation whereby each autonomic element is working towards its own goal but, the overall set of actions/policies degrades drastically the performance and dependability of the network. Such a situation could even result in unwanted oscillations and instabilities in the control-loops. These issues raise the question of how stable and efficient are the control-loops of an autonomic node or network. Therefore, methodologies and mechanisms are required for synchronizing the actions undertaken by diverse autonomic elements in a self-managing node and the network as a whole. One applicable approach is the use of Formal Description Techniques (FDTs) during the design of an autonomic node or

network [2]. On the other hand, the approach that is presented in this paper aims at ensuring the stability of control-loops during the run-time/operation of an autonomic node/network.

The GANA *(Generic Autonomic Network Architecture)* [1] is evolving in the course of the EFIPSANS [3] project. We propose to extend the GANA architecture in a way ensuring that the set of actions/policies issued by diverse autonomic entities in a particular time slice are analyzed such that only those actions that contribute to the overall global goal of the network, are allowed to be executed.

The rest of this paper is organized as follows: Section 2 deals with the architectural extensions to GANA for the purposes of addressing run-time stability of control-loops of an autonomic node and network. Section 3 defines an optimization problem that is at the heart of our approach. Section 4 provides the results of some experiments that were conducted in order to evaluate of our approach. Finally, section 6 draws conclusions and outlines some future research directions.

2. Extending the GANA in order to address Control Loop Stability

The *GANA* specifies a *control loop* by introducing two basic concepts – a *Managed Entity (ME)* and a *Decision Element (DE)*. Conceptually a DE is an autonomic element responsible for the management of concrete MEs (e.g. protocol modules), and realizes a control loop based on its continuous learning cycle. Through a specified control loop a DE is able to exercise specific algorithmic schemes and/or policies on its ME(s). Taking into account that control loops can be implemented at different levels of abstraction – e.g. on node or network level, GANA provides the *Hierarchical Control Loops (HCLs)* framework consisting of four levels at which DEs and associated control loops can be introduced – *Protocol Level* (e.g. a control loop inside a single protocol), *Functions Level* (in-built management of basic networking functions such as Forwarding), *Node Level* and *Network Level*. Thereby DEs on a higher level manage DEs on a lower level. For more information we refer the reader to [1].

In this work, we borrow some ideas from the area of Control Theory and apply them inside the GANA architecture. Firstly, the layered structure of the *HCL* should be used for implementing stability hierarchically. Secondly, every set of DEs which have some overlapping and possible contradictions in the actions, policies or configurations they are responsible for, should refer to an arbiter that allows only those management actions to be executed, which are beneficiary for the overall fitness of the network. We call this arbiter an **Actions Synchronization Module (ASM)**. An ASM is considered as part of a dedicated DE that has been elected or is by design the most appropriate one for acting as an arbiter (enabling the negotiation over the tentative actions). We consider plain actions, policy enforcements, and the deployment of complex configuration profiles as (management) actions. Therefore, we require that every DE should keep a list (catalogue) of the actions it is allowed to issue without having to consult an upper level DE. As illustrated in Figure 1, if a DE (e.g. DE1.1, DE1.2 or DE1.3) faces a problem that is beyond its local scope, i.e. the action to be issued as a response to some challenging conditions is not inside the aforementioned catalogue, it should consult its upper level DE (DE2.1 or DE2.2). The

upper level DE should in turn consult the corresponding ASM (hosted by DE2.2) that is expected to solve an optimization problem and to respond back with a set of actions that are allowed to be executed.

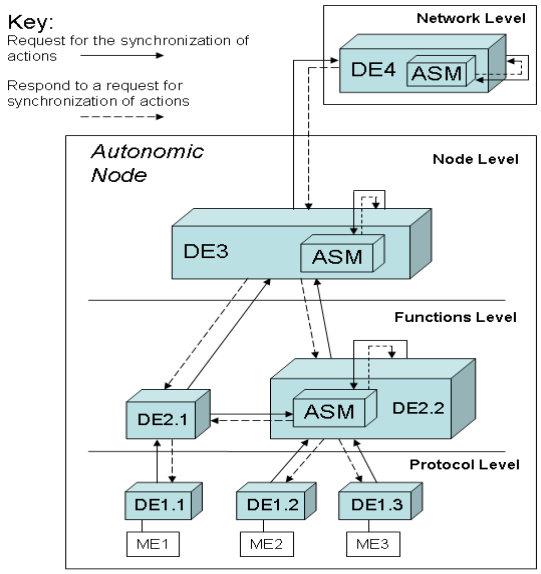

Figure 1: Hierarchically addressing stability within GANA

After the allowed actions have been selected, the upper level DE informs the lower level DEs whether or not they are allowed to "fire" some of the actions on their corresponding MEs (ME1, ME2, and ME3). On the other hand if the upper level DE "recognizes" that the actions it has been requested to synchronize are beyond its competence, it should further consult its corresponding upper DE (e.g. DE3 in Figure 1) on the higher level.

3. A Binary Integer Program for addressing run-time Stability in GANA

In this section we define an optimization problem that should be solved by an ASM whenever a set of actions needs to be synchronized.

Let Q be a set of independent QoS metrics ($|Q| = n \in N^+$). *Independent* in this case means that the metrics do not have co-relations with each other. Let W be a set of weights, each of which is assigned to one of the metrics contained in Q ($|W| = n \in N^+$, $w_i \in W$, $w_i \in R^+, i \in \{1, ..., n\}$). These weights represent the importance of each metric for the overall health of the network. Considering a particular point in time t_0, the values of the metrics in Q are represented by a vector $Q(t_0)=(q_1(t_0), ..., q_n(t_0))^T \in R^n$. Suppose that the higher the values of $q_1(t_0), ..., q_n(t_0)$ the better the performance of the system (for metrics which require to be minimized one can take the reciprocal value), then the following expression gives the network fitness at t_0.

$$NF(t_0) = \sum_{i=1}^{n} w_i q_i(t_0) = \langle w, Q(t_0) \rangle \quad (4.1)$$

Hence the goal of the autonomic mechanisms in the node/device and the network is to maximize *NF(t)* throughout the operation of the system. Additionally let A ($|A|=M \in N^+$) be the set of actions that the ASM is responsible for. By $a_j \in A$ with $j \in \{1, ..., M\}$ we denote a single action. Further, the domain relation of an action $d: A \rightarrow \{0,1\}^n$ is introduced. The relation takes as an input an action and returns a (0-1) binary vector. If the i^{th} component of the vector is 1 then the i^{th} metric is influenced by the input action, and respectively if its 0 then the metric is not influenced. We define the *domain matrix* of A as $D = (d(a_1) \cdots d(a_M)) \in \{0,1\}^{n \times M}$. Further, we introduce the impact relation of $A - I: Q \times A \rightarrow R$. The output value of *I(i,j)* stands for the impact the j^{th} action has on the i^{th} metric. Thus, if only action a_j is executed at point in time t_0, then the new value of q_i will be $q_i(t_0)+I(i,j)$. Based on the definitions given above, and assuming that in a particular time slice a total number of $m \in N^+$ ($m \leq M$) actions have been requested for synchronization, the following optimization problem is defined

$$p = \arg\max_{p \in \{0,1\}^m} \sum_{i=1}^{n} w_i (q_i(t_0) + \sum_{j=1}^{m} p_j I(i,j)) = \arg\max_{p \in \{0,1\}^m} w^T I_m p + w^T Q(t_0) \quad (4.2)$$

$$s.t. \quad D_m p \leq c$$

I_m stands for an $n \times m$ real-valued matrix where $I_m(i,j)$ *re*presents the impact of the requested j^{th} action on the i^{th} metric. Moreover, in (4.2), p stands for a (0-1) vector which gives whether a particular action has been allowed to be executed or not. If the j^{th} position of p is 1 then the corresponding action has been selected for execution, otherwise it has to be dropped. Hence, an optimization with respect to p is equivalent to selecting the optimal set of actions requested in a particular time frame. Moreover, the matrix D_m is a sub-matrix of the domain matrix D of A consisting only of the columns which represent the domains of the requested actions. The vector $c \in N^n$ determines the extent to which actions with overlapping domains are allowed. For example, if $n=4$ and $c=(1,1,1,1)^T$, i.e. only four QoS metrics are considered, then the additional constraint says that only one action influencing a single metric is allowed. That is, the additional constraint can be used to enforce the resolution of conflicts between actions with overlapping domain regions.

Since the term $w^T Q(t_0)$ in (4.2) is just a constant that reflects the current state of the network, it can be dropped, which is very good news because it means that the values of the QoS metrics are not needed for the optimization. Thus, no interaction with the monitoring functions measuring the metrics is needed. The final optimization problem takes the following form:

$$p = \arg\max_{p \in \{0,1\}^m} w^T I_m p$$

$$s.t. \quad D_m p \leq c$$

While we are aware of the fact that linear mixed integer programs are NP-hard in general, modern solvers are quite advanced and would provide a solution that is the best possible based on the underlying optimization algorithm.

4. Evaluations

In order to conduct some initial evaluations of the proposed methodology, we implemented an ASM on a Linux Ubuntu Machine. The prototype is based on Multi-Threading and the Coin-OR solver [4][5].The focus of the evaluations was set on the issue of overhead produced by solving the previously derived optimization problem.

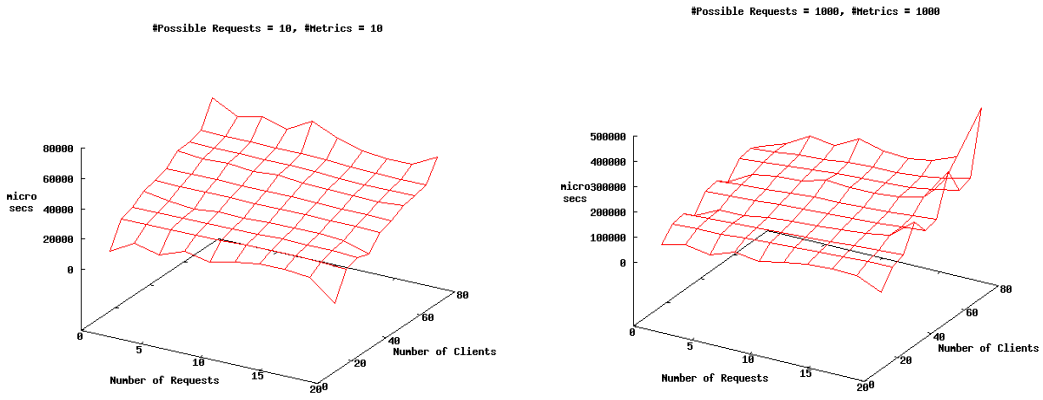

Figure 2: Performance evaluations of the proposed approach

The evaluations were conducted on a single machine - Intel(R) Pentium(R) M processor 1.60GH, 509.2 MB RAM. Our goal was to examine the performance of the overall processes implemented by the ASM component. Thus several experiments were carried out with different numbers of metrics and actions, different numbers of clients simultaneously requesting for synchronization, and different number of requests issued by every client. The values of the impact matrices and the weights reflecting the significance of the metrics were randomly generated for each experiment. Figure 2 presents the results of our performance evaluations. The time measurements that are plotted on the Z-axis correspond to the maximum response time of an ASM component, after each of the simultaneously started ASM clients have issued a number of requests (shown on the X-axis). In the above described environment it was impossible to complete the experiments with more than 70 clients started simultaneously. This could be due to the prototype design or an overload of the operating system. Despite this drawback, one can observe (Figure 2) that for 70 and fewer clients the response times were quite reasonable. Hence, depending on the number of the possible and relevant requests, we deduce that one could use this approach presented here for both fast decisions, e.g. on routing, forwarding level or decisions related to the resilience of the system, and for synchronisations which are not strongly time constrained.

5. Conclusions and Future Work

This paper proposed an approach for addressing run-time stability in the context of the GANA. GANA is a recently emerged Reference Model for Autonomic Network

engineering. However, the methodology proposed here is applicable not only to GANA but also to any other architectural model for autonomic systems engineering. The idea presented here is to introduce special components called Actions Synchronization Module (ASM), which should be part of a Decision Element (DE). DEs are responsible for managing their assigned Managed Entities (MEs). The aforementioned ASMs provide the functionality of "gathering" (from the DEs) management actions, policies and configuration profiles to be deployed, and synchronizing them based on a binary linear program that selects the best subset of actions allowed to execute at a particular time slice. By doing that issues such as contradicting actions and policies, oscillations and chaos caused by various DEs trying to optimize the performance of their own MEs, without considering the overall network fitness are avoided.

Our evaluations showed that the prototype is applicable inside a GANA node and scales well (for the purposes of GANA) with respect to the number of clients as well as the size of the requests and the optimization problem parameters. Thus, our methodology could be applied for tasks with real-time requirements such as improving the resilience of the autonomic systems.

In general, we conclude that it is important to apply all the methodologies that guarantee stability of control-loops, ranging from the application of model-driven/FDTs-driven analysis, simulation and validation of control-loop structures and behaviors at design phase, to the application of optimization techniques at run-time.

Our future work in that direction will mainly target the application of the presented methodology in a real GANA network, as the one that is developed in the course of EFIPSANS [3]. As a short paper focusing mainly on the importance and applicability of Synchronization techniques in addressing some aspects of stability of interacting control-loops, we have left out the illustration with real-world scenarios (a subject to be covered by the future long-version of this paper). Also as part of further work, scalability issues will be further examined and addressed.

Acknowledgement: This work has been partially supported by EC FP7 EFIPSANS project (INFSO-ICT-215549).

References

1. R. Chaparadza: Requirements for a Generic Autonomic Network Architecture (GANA), suitable for Standardizable Autonomic Behavior Specifications for Diverse Networking Environments, IEC Annual Review of Communications Volume 61, December 2008
2. Jia Bai et. al.: A Model Integrated Framework for Designing Self-managing Computing Systems, in the proceedings of FeBid 2008
3. EC funded- FP7-EFIPSANS Project: http://efipsans.org/
4. Stuart R. Thorncraft: Evaluation of Open-Source LP Optimization Codes in Solving Electricity Spot Market Optimization Problems, in the proceedings of ORMMES 2006
5. Coin-OR: http://www.coin-or.org/ as of date 01.10.2009