Addressing Stability in Future Autonomic Networking

Timotheos Kastrinogiannis[1], Nikolay Tcholtchev[2], Arun Prakash[2], Ranganai Chaparadza[2], Vassilios Kaldanis[3], Hakan Coskun[2], and Symeon Papavassiliou[1]

[1] Institute of Communications and Computer Systems (ICCS), School of Electrical and Computer Engineering, National Technical University of Athens, Athens 15780, Greece
timothe@netmode.ntua.gr, papavass@mail.ntua.gr
[2] Fraunhofer FOKUS Institute for Open Communication Systems, Berlin, Germany
{nikolay.tcholtchev,arun.prakash,ranganai.chaparadza, hakan.coskun}@fokus.fraunhofer.de
[3] VELTI S.A. – Mobile Marketing & Advertising, Athens, Greece
vkaldanis@velti.com

Abstract. When considering autonomic networking, where multiple self-* functionalities, in terms of node-wide or network-wide control loops, must operate, interact and proficiently collaborate, stability problems inherently arise due to the distributed nature of the decision making process and autonomic nodes interactions towards enabling various self-* functionalities, along with the stochastic nature of the networking environment. This article provides a systematic, concrete view of stability in autonomic networks design. It aims at identifying and categorizing fundamental autonomic networks' architectural and designing issues that cause or affect stability, highlighting and discussing corresponding solutions and thus, providing theoretic tools for analyzing and treating them. As a reference model we adopt Generic Autonomic Network Architecture (GANA), a holistic framework for autonomic networks engineering.

Keywords: Autonomic Networks, Stability, Control Loops.

1 Introduction

The vision of Future Internet, is of a self-managing network whose nodes/devices are designed/engineered in such a way that all the so-called traditional network management functions, i.e. FCAPS framework (Fault, Configuration, Accounting, Performance and Security) [1], as well as the fundamental network functions (e.g. routing, forwarding, monitoring, mobility, resource management, e.t.c.) are enhanced with autonomic attributes. In such an evolving environment, it is envisioned the network itself to detect, diagnose and repair failures, as well as to constantly adapt its configuration and operations aiming at optimizing its performance.

In general, towards realizing networks' autonomic behaviors, the presence of control-loops in the system is essential. Inputs to a control loop consist of various status signals, information and views continuously exposed from the system, component(s) or resource(s) being controlled (e.g. protocols, nodes, functionalities, etc.), along with (usually policy-driven) management rules that orchestrate the behaviour of the system or component(s). Outputs are commands to the system or component(s) to adjust its operation, along with status to other autonomic systems.

K. Pentikousis et al. (Eds.): MONAMI 2010, LNICST 68, pp. 50–61, 2011.

Practically, control loops consist of iterative and (most of the time) distributed algorithms, that enable various node's self-* behaviours and hence, guide them to act in line with their own optimization goals or towards achieving global optimal network objectives. Henceforth, future autonomics envisions the aggregation of node-scoped control loops, i.e. within a node, in terms of interacting intra/inter-node control loops or triggered/managed low level control loops by higher level control loops within the node or the network. Intuitively, the above view leads to a hierarchal control loops paradigm that enables the efficient design of autonomic nodes and architectures [2].

As the designers of a network's architecture increase its autonomic attributes, in terms of introducing various self-* functionalities (i.e. control loops) at node or network level, the inherent issue of stability becomes more and more complex and crucial. In general, the stability of dynamic systems has been extensively studied since the beginning of *Control Theory*, but when viewing it through a dynamic networking environment, additional drawbacks and challenges emerge [3].

In most cases, stability is defined as a property of a (dynamic) system or element through which it is reassured (it reassures) that its output will ultimately attain a steady state (i.e. the state when the recently observed behaviour of the system will continue into the future). When considering an autonomic networking environment, where multiple self-* functionalities must operate, interact and collaborate, in terms of node or network wide control loops, its stability implies reaching an equilibrium point over a finite time frame. In other words, the system/network should be stable in the sense that parameter values change, however remaining bounded in a small neighbourhood of a final value. This is especially important since self-* algorithms mainly run completely automatic and without the possibility of manual intervention.

Despite the vital role of stability in future autonomic networking, there is a lack of a concrete framework and corresponding theoretic and designing tools for addressing and treating autonomic networks' stability. Recent attempts to address the latter are either closely coupled to the studied networking environment [4] or the autonomic functionality that is engineered [5]. Towards enlightening the evolutionary path of future autonomics via highlighting the significance and role of stability in such dynamic networking paradigms, this paper makes the subsequent contributions:

- Identifies, categorizes and discusses the key factors that affect autonomic networks' stability attributes from both theoretic and designing point of view.
- For each one, corresponding theoretic tools and concrete methodologies for studying and treating them are provided.
- Proposes a concrete framework for addressing both off-line (network designing phase) and on-line (network runtime phase) aspects of stability.

The rest of this paper is organized as follows. In section 2, fundamental designing principles and the GANA model are highlighted. Section 3 highlights the key principles of the proposed methodology for addressing stability in autonomic networks. Section 4, identifies, analyses and provides solutions to fundamental autonomic networks' stability issues, concerning both theoretical and architectural aspects at design time. Then, section 5, illustrates the concepts of action synchronization functions and self-stabilization monitoring towards tackling the issue of runtime autonomic networks' stability. Finally, section 6 concludes the paper.

2 An Overview of the Generic Autonomic Network Architecture

The concept of "autonomicity" – realized through control-loop structures operating within network nodes/devices and the network as a whole, is an enabler for advanced and enriched self-manageability of network devices and networks. To address the lack of such autonomic notions formalism, in both theoretic as well as pragmatic designing levels, an evolvable, standardizable architectural Reference Model for Autonomic Networking and Self-Management within node/device and network architectures, dubbed the Generic Autonomic Network Architecture (GANA) [2], has recently emerged. The central concept of GANA is that of an autonomic Decision-Making-Element ("DME" or simply "DE" in short—for Decision Element). A Decision Element (DE) implements the logic that drives a control-loop over the "management interfaces" of its assigned Managed Entities (MEs). Therefore, in GANA, self-* functionalities such as self-configuration, self-healing, self-optimization, etc, are implemented via a Decision Element(s).

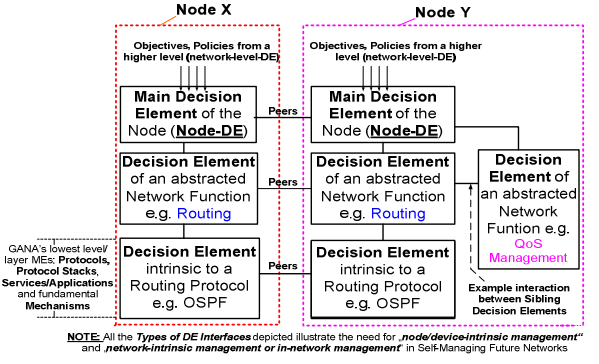

Fig. 1. Illustration of hierarchical, peering, sibling relations and interfaces of DEs in GANA

Specifically, GANA introduces "autonomic manager components" i.e. Decision-Elements (DEs), meant to operate at four different abstraction levels of functionality. These "autonomic manager components" are designed following the principles of "hierarchical", "peering", and "sibling" relationships among each other within a node or network. GANA defines four hierarchical levels of abstractions for which DEs, MEs, Control-Loops and their associated dynamic adaptive behaviours can be designed, capturing a "holistic view of interworking autonomic and self-management levels". Thus, the levels of (DEs) abstractions are as follows:

Level-1: protocol-level (the lowest level) by which self-management is intrinsically designed and associated with a network protocol itself (whether monolithic or modular) [6], [7].
Level-2: the abstracted function-level (directly above the protocol(s)-level) that abstracts some protocols and mechanisms associated with a particular function e.g. "routing function", "forwarding function", "mobility management function", etc.
Level-3: the level of the node/device's overall functionality and behaviour i.e. a node or system as a whole, is also considered as level of self-management functionality.

Level-4: the level of the network's overall functionality and behaviour (the highest level). Network-Level-DEs are characterized by the following attributes: (1) they are the ones with wider network-wide view to perform sophisticated decisions e.g. network optimization; (2) they are logically centralized to avoid processing overhead in managed nodes with having distributed decision logic in network elements; (3) they are the ones that provide an interface for a human to define goals and objectives or policies e.g. business goals to the autonomic network.

Fig. 1 illustrates a GANA aware autonomic node, whereby lower level Decision-Making-Elements, operating at the level of abstracted networking functions, become the Managed-Entity of the upper layer Decision-Making-Element of the system (node), exposing "views" to the upper level DME(s), which uses its overall knowledge to influence (enforce) its lower level DMEs to take certain desired decisions, formalizing the notion of DEs hierarchy.

In the following sections, without loss in the generality of the proposed methodology, we address and study stability in autonomic networks based on the concepts, design principles and attributes of the GANA framework.

3 Stability Issues in Autonomic Networking

There are two key complementary notions of stability when designing autonomic network architectures namely, autonomic node's stability and autonomic network's stability. Node stability concerns the interactions of control loops (self-* functionalities) that exist within the same node, either in the same or different layers of its protocols' stack or concerning various networking functionalities. Network stability copes with the interactions among control loops located in different components of the network; therefore, a two steps approach is required, i.e.

- Initially, the proficient collaboration of control loops towards reaching a stable outcome must be reassured without considering the drawbacks imposed by the networking environment (i.e. assuming a deterministic environment).
- Then, the same study should be repeated when the proposed autonomic approach (e.g. mechanism, architecture, etc) functions/operates over the actual network. At that point, additional challenges emerge that may affect or prevent the efficient collaboration of control loops, like the presence of churn, the loss of communication between nodes (and their corresponding control loops) or even, the existence of miss-behavioural or malicious autonomic nodes.

In either of the previous cases, a methodological approach is required when addressing and studying stability in an autonomic networking paradigm, placing special emphasis on the communication and collaboration among the introduced control loops, regarding their managing, managed, peering and sibling relationships. Specifically, the following key issues must be considered:

Issue I. Self-* functionalities (i.e. control loops) interactions and their convergence at a stable outcome (i.e. equilibrium point). Thus, it is vital to reassure that only important changes cause the triggering of actions or the recalculation of parameters (in order to reduce the causes of instability). Towards this direction, we introduce and

exploit well established guidelines that stem from concepts in the fields of Game Theory and Optimization Theory towards:

- Establishing well-defined valid operating regions of particular control loops.
- Decoupling control systems by ensuring that they control different independent outputs based on independent inputs and if this is not possible, then tuning them so that they impose control at very different timescales. Such an approach will allow us to decouple systems that would otherwise be strongly coupled.

Issue II. Conflicts resolution, not only between self-* functionalities and behaviours (i.e. control loops) belonging to different network components but also among control loops within the same node. Conflicts may occur when more than one control loops manage or affect the same functionalities or resources, especially when they are placed in a hierarchical manner.

Issue III. Time scaling issues among collaborating control loops. In this case, the main difficulty is that timing varies significantly when considering various self-* functionalities which are interacting, while residing on different layers of a node's/network's protocols stack. Thus, it must be reassured that the following two dynamic and interacting sets of "events", that is {signalling, monitoring information and decision making} and {changing environment and triggering events}, are changing on a similar timescale.

In the following we analyze each of the above critical issues and we provide concrete methodologies and key theoretic tools for addressing and treating them.

4 Designing for Stability

In this section we place emphasis on revealing vital aspects related to stability (i.e. control loops stability) that need to be addressed while still being at the design time of an autonomic network and its corresponding autonomic elements, such as the GANA DEs. Hence, we propose solutions and corresponding theoretic tools to address the latter, in contrast to aspects of stability that must be handled at run-time (i.e. as the autonomic system evolves) which are highlighted in the next section.

4.1 Stable Autonomic Behaviours Design through Game Theory – *From Theory to Theory*

Despite the variety of alternative autonomic architectures that may emerge when obtaining the solution of the corresponding initial system's optimization problem, one common designing attribute characterizes them, the decentralized nature of the consequent autonomic algorithms/mechanisms. This not only necessitates the collaboration of various network components to achieve different layering objectives, but also implies the distribution of the decision making procedures of the network among its components, instead of traditional centralized approaches, which eventually increase the role of individual network components.

Aiming at composing efficient distributed and iterative autonomic algorithms, appropriate designing theoretic tools need to be adopted, which are promoting

network's autonomic nature. To that end, Game Theory and Network Utility Maximization (NUM) theory via its decomposition methods [8] (i.e. a theoretical tool for designing optimal distributed algorithms over layered architectures) consist of some of the most widely adopted mathematical frameworks applied in autonomic networking. On one hand, these analytical tools can be used to derive distributed algorithms and determine their efficient collaboration (i.e. autonomic node's DEs (i.e. control loops) signalling), providing theoretically founded answers to the question: *who does what in the autonomic network and how to connect them.* On the other hand, they inherently provide methodologies not only for investigating and reassuring that the corresponding derived distributed algorithms are reaching a stable outcome, in terms of equilibrium points, but also for enabling the following desirable designing attributes: a) limited amount of overhead among nodes (and control lops), b) fully asynchronous updates and c) robustness to arbitrary signaling information [9].

In general, the following thread of analysis holds. An autonomic network's/ functionality's stability is assured via (a) DEs cooperation stability, which can be further reassured via (b) their corresponding control loops collaboration stability and thus, via (c) proving the stability of the distributed iterative algorithms that steer autonomic node's control loops, obtained via the solution of the corresponding optimization problems.

In the following section, we place emphasis and analytically present a concrete methodology for studying autonomic mechanisms' stability via game theory.

How to Treat Stability via Analytical Methods? - A Game Theoretic Approach

Game theory is currently widely explored in autonomic networking due to the following two main inherent attributes:

i) Autonomic nodes' selfish non-cooperative nature can be properly defined and formulated as a non-cooperative game, where nodes act as individual players aiming at maximizing their own interests. Then, game theory provides the foundations and mathematical tools for studying and solving such problems.

ii) The distributed nature of the produced algorithms, which seek non-cooperative game's solution and steer corresponding autonomic nodes DEs (i.e. are their control loops), facilitate the efficient design of a network's autonomic mechanisms.

A non-cooperative game consists of three basic components, (i) a set of players S, consisting of N autonomic nodes, (ii) a set of actions A_i, i.e. the feasible strategy space of i autonomic node and (iii) a set of preferences, which can be expressed via appropriately defined utilities $U_i(a)$ of each autonomic node, where $a_i \in A_i$. Thus, the corresponding non-cooperative game is formulated as follows:

$$\max_{a_i \in A_i} U_i(a_i) \quad s.t. \ a_i \in A_i \ . \tag{1}$$

Furthermore, regardless of the designed problem's formulation, a non-cooperative game's investigation requires the consideration and analysis of the following key steps:

A. Game's steady state via the existence of equilibriums (e.g. ***Existence of Nash equilibrium***).
B. Equilibrium's properties (e.g. ***Nash Equilibrium's properties***).
C. Optimality of equilibrium (e.g. ***Pareto optimality***)

D. Convergence of equilibrium via studying the convergence of users' corresponding (best) response towards selfishly maximizing their utilities, thus reaching game's equilibrium. (e.g. ***Convergence of Nash equilibrium***).

How to address stability via Game Theory? Steps A and D can provide the answer to the previous question. Specifically, upon setting a non-cooperative game, with respect to the required behaviour of the autonomic nodes in the network, and upon deriving distributed algorithms that steer nodes to have the expected behaviour (i.e. autonomic node's control loops), the ability of the corresponding autonomic network to reach a stable outcome (i.e. an equilibrium point) can be analytically proved:

1. By showing the existence (or even the uniqueness) of such a stable point (**step A**);
2. Via proving that the derived distributed (and often iterative) algorithms (i.e. control loops) will always reach such a point **(step D**).

Intuitively, that latter suggests that autonomic nodes' interactions, via their corresponding DEs (control loops), will always reach a stable outcome; thus, leading to autonomic network stability, under the assumption that DEs communication synchronization is always achieved. Game theoretic tools that allow treating issues A and D are super-modular games, utility functions' properties (e.g. quasi-concavity), potential games, coalition games e.t.c. Moving one step forward, the stochastic nature of the networking environment should be considered (i.e. dropping the assumption of reassured DEs communication and synchronization). This makes the analysis of the corresponding games much more difficult. To that end, stochastic games formulations can be applied. A deeper analysis on game theory in autonomic networks is out of this paper's scope; interested readers are referred to [10] and [11].

4.2 Addressing Stability in an Architectural Level – From Theory to Practice

Apart from treating stability via analytical theoretic means, in the following we identify a plethora of vital autonomic network architectures' designing aspects that play a crucial role in determining their stability attributes. Throughout our analysis, the benefits of adopting GANA [2] for devising stable autonomic networks are also revealed, since GANA inherently adopts (i.e. inherent architecture's attributes) the key prerequisites illustrated below.

4.2.1 Hierarchy of Control-Loops (DEs)

An important feature of an autonomic architecture is to maintain a hierarchy of control-loops (as defined in GANA). The benefits from introducing hierarchy to manage complex autonomic systems are extensively justified in [3]. From an autonomic network's stability point of view, the introduction of hierarchy allows the horizontal (among network nodes) and/or vertical (different functional levels) partition of the decision making process. Thus, the failure of a single entity or DE will not result in the total failure of the network under control. DEs with independent goals and policies will continue to operate and manage their corresponding protocols, allowing the failed DE to restart or "correct" itself in the mean time. This feature of GANA allows the network (or at least part of functionalities) to be stable in spite of the failure of a single DE or few DEs.

In addition, some of the major drawbacks of traditional hierarchical systems are large convergence time, sensitivity to failures, large computational power requirement and communication overhead. GANA on the other hand does not follow the traditional hierarchical systems architecture; i.e. it's distributed hierarchical system architecture. Thus, it allows communication between sibling and peer DEs, providing a distributed solution to control its MEs.

To that end, GANA incorporates the following key design principles: **(1)** Lower level DEs expose "views" up the Decision Plane, allowing the upper (slower) control loops to control the lower level (faster) control-loops driven by lower level DEs); **(2)** Changes computed in the upper DEs implementing slower (i.e. within larger time frames) Control-Loops are propagated down the DE hierarchy to the Functions-Level DE(s) implementing the faster control-loops that then arbitrate and enforce the changes to the lowest level Managed Entities (protocols and mechanisms). This operation reduces the drawbacks of large convergence and sensitivity times, traditionally found in conventional hierarchical systems. Finally, the nature of the distribution of the tasks to DEs inside the node and Network-Level DEs further ensures that the communication and computation power requirements are kept to a bare minimum inside the node.

4.2.2 Concept of "Ownership"

The "Concept of Ownership" is another feature of the intrinsic stability attributes that must be considered for an autonomic network architecture (as is also defined in GANA). This concept requires that every ME is managed by a single DE, i.e. no two DEs (i.e. control loops) can control the same ME (i.e. functionality, resource, e.t.c.) at any given point of time in the network. This is important from system's stability point of view since it relieves the burden of "*conflicts resolution*". Specifically, if an ME is controlled by two or more DEs at the same time, contrasting, conflicting and at times repetitive policies, objectives and reconfiguration requests, etc, originating from different DEs lead to an unstable ME and thus, to an unstable autonomic network. Through the "Concept of Ownership", GANA ensures that this instability is avoided.

4.2.3 Separation of "Operating Regions"

Another prerequisite of an autonomic architecture is the efficient separation of the "operating regions" for the control-loops as advocated in [12]. This can be achieved by decoupling control systems and ensuring that they control different independent outputs based on independent inputs, as defined in GANA. If it is impossible to decouple certain outputs and inputs from affecting each other, it is important to reassure that they impose control at different timescales on their MEs.

4.3 Model-Based Techniques

Model-driven approaches for design DEs and their inter-relationships should also be exploited to efficiently address aspects related to stability of control-loops. Specifically modelling and validation of DEs autonomic behaviours using Formal Description Techniques (FDTs), such as the well-known and successful ITU-T SDL language, can be explored to address certain aspects of stability. Such an approach would enable the design, model-checking and verification of DEs, as well as the validation, simulation and some partial code-generation of autonomic behaviours of

DEs. Finally, the aspects (i.e. methodology) [13] that need to be taken into account when following such an alternative are: (1) A Meta-Model that enforces constraints in the design models for DEs and their interactions with assigned MEs and with other DEs, that should be embedded in the Modelling Tools e.g. a "model editor", is required; (2) A Model-Walker that can be designed for walking over a design model in order to detect conflicting control-loops and overlaps in the so-called "valid operating regions" of control-loops specific to DEs; (3) Simulations for detecting behaviour conflicts between interacting control-loops designed to operate within a node/device and in the network.

5 Addressing Stability at Runtime

After applying the methodologies and guidelines presented in the previous sections, an autonomic network is intrinsically equipped with interacting control loops having some degree of self-stabilization. However, it is possible that due to the stochastic nature of the networking environment, situations may occur, which have not explicitly been considered during the design time of the Decision Elements. We denote such situations as *"emergent situations/behaviours"*. Intuitively, "emergent behaviours" can be considered as an indirect interference between a set of DEs, whereby each DE is optimizing its own goal based on its embedded logic, but the overall set of executed actions is leading the system to an unstable state of oscillations and continuous responses of diverse control loops. In order to avoid such emergent behaviours, the concept of Action Synchronization Functions (ASFs) has been proposed in [14] (which is also part of a GANA's Decision Elements). In simple words, [14] introduces ASFs in the GANA hierarchical structure, in order to allow DEs on a lower level to resolve potential conflicts via requesting DEs belonging to a higher level (in the DE's hierarchy) for taking over their synchronization and coordination. Specifically, after a number of DEs have referred to a higher level DE in order to be informed whether particular tentative action(s) are allowed or not, the higher level DE selects the optimal subset of tentative actions reported by the corresponding lower level DEs. The tentative actions are gathered over a pre-defined time window. Consequently, the higher level DE responds back to the requesting lower level DEs on whether they are allowed to proceed with executing tentative actions or not.

There are various aspects of stability which are addressed by the architectural role of the ASF namely, 1) acting as an arbiter for conflict resolution among DEs with potentially interfering actions, 2) exploiting the GANA hierarchical structure in order to realize the notion of hierarchical optimal control – autonomic entities on a higher level have access to more global information, and 3) enabling the gathering of actions for synchronization over a pre-defined time period has a smoothing effect on the rate at which control loops operate, thereby avoiding sudden oscillations and chaotic situations in the autonomic network.

In [14], the issue of conflict resolution is tackled via deriving a binary integer program, which aims at the optimization of a set of key performance indicators (KPI) by selecting an optimal subset from the $m \in N^+$ actions waiting to be synchronized:

$$\max_{p \in \{0,1\}^m} w^T I_m p \quad \text{s.t.} \quad D_m p \leq c \tag{2}$$

where, *w,* is a vector containing a weight value for each considered KPI. These weights specify the importance of each KPI for the network and are specified by the network operator; *P,* is a binary (0-1) vector being optimized. This vector reflects the actions that have to be synchronized. A "1" at a particular position means that the corresponding action has to be executed. A "0" stands correspondingly for the case when an action must be stopped. I_m is an impact matrix (similar to a payoff matrix in the context of game theory) that specifies the impact of the tentative actions on the KPIs considered by the ASF.

The constraints of the previous optimization problem are specifying the number of actions that are allowed to simultaneously influence (as specified in I_m) a particular KPI. The matrix multiplication on the left side of the inequality gives the number of actions which influence the KPIs. In order to achieve that, D_m is defined as a binary (0-1) matrix, in which a "1" is set in case $(I_m)_{i,j}>0$ and a "0" in case $(I_m)_{i,j}=0$.The components of the vector $c \in N^m$ stand for the number of actions which can simultaneously influence the corresponding KPI.

We can interpret the above optimization problem as *"selecting the most appropriate subset of tentative actions such that the change in the state of the system is positively maximized"*. For a bottom-up definition of this optimization problem, starting with the current state (as defined by the current KPI values) of the system, we refer the reader to [14]. Unfortunately, binary program (2) belongs to a class of NP-hard problems. For that purpose, in this work, we propose to relax the binary constraint imposed on the vector *p,* and turn it into an inequality: $0 \leq p_i \leq 1,\ i \in \{1, \ldots, m\}$. Thus, the new optimization problem (contrasted to the one in [14]) is given by:

$$\max_{p} \ w^T I_m p \quad \text{s.t.} \ D_m p \leq c,\ 0 \leq p_i \leq 1,\ i \in \{1, \ldots, m\}. \tag{3}$$

This optimization problem is a linear program which constitutes a convex optimization problem. Hence, there is only one optimum and every local optimal solution is also a global one, which means that the diverse optimization techniques will always improve iteratively the quality of the obtained solution. The above formulation is non-NP-hard. The result of this optimization is a vector containing values from the interval *[0, 1]*. If *ith* component of this vector is 0 then the corresponding action is disallowed. Correspondingly, if it is 1 then the action should be issued. If a DE, requesting for synchronization, receives as response a value from the interval *(0, 1)*, then it can either execute or drop the action, based on an internal threshold θ_i. These thresholds can be supplied by the human experts tweaking the autonomic network. For instance, the history of requests for synchronization can be analyzed offline (e.g. by employing statistical and/or machine learning methods) and appropriate thresholds can be extracted using some statistical or optimization tools.

5.1 Autonomic-Aware Metrics to Infer and Self-assess Stability

Since an autonomic network is expected to be self-adaptive to changes and challenges in its environment during its operation, it must be able to self-assess and infer stability related problems experienced at various levels of the Decision Plane Hierarchy. The assessment of stability at different levels where a DE implements an autonomic functionality (e.g. autonomic routing, autonomic QoS-Management, autonomic Mobility-Management, etc), requires that every DE must implement some monitoring

functionalities towards self-awareness (i.e. implement *Counters* for storing statistics e.g. i) the number of times a DE enforced a change the behaviour of its assigned Managed Entities (MEs) in reaction to specific events over a period of time, or ii) the number of times a DE received information indicative of instability problem from its MEs). What is also required, are *Timing Variables* (e.g. timers) for storing time durations that measure some DE activities. Diverse types of Timing Variables are required for each type of input that flows into a DE—according to the "valid operating region" of its associated control-loop that was captured and defined at design time.

Thus, every DE must expose the "views" captured by the *Counters* and the *Timing Variables* to its upper DE, which then assesses the "*degree of stability*" of the autonomic functionality realized by the particular lower level DE and decides to enable an appropriate response strategy towards reassuring stability. The upper DE may further aggregate some statistics and "views" and expose then further up the Decision Plane (possibly up to the level of network-level DEs) where more sophisticated decisions can be taken based on the wider knowledge about the network that is gathered by network-level DEs. Finally, an autonomic behaviour triggered by a DE reacting to a change may inductively span a number of DEs, and their associated control-loops, which are enabling (via collaborations and interactions) *an ultimate goal* (i.e. an autonomic behaviour). Such an *ultimate goal* could be the (re)-setting of a parameter value on a single ME or multiple parameters on an ME(s) managed by the first triggering DE in the DE interaction chain, or could be the setting of multiple parameters on ME(s) managed by one or more other DEs involved in the DE interaction chain. A DE e.g. a Node-DE, belonging to a higher level than the DEs involved (which could be Function-Level DEs), provided it knows the *causality graph for actions/policies employed by different DEs* to achieve the *ultimate goal*, could then use information stored in *Timer* variables stored by lower-level DEs in the interaction chain, in order to infer stabilization time and challenges, via using this knowledge to improve cognitive response strategies over time.

6 Concluding Remarks

In this paper we address the issue of autonomic networks stability. Despite the existence of recent elegant analytic results in specific networking paradigms, the issue of stability in autonomic networking still lacks of concreteness, generality and mainly extensive validation; hardening the wide applicability of autonomic solutions in realistic environments. Following a pragmatic and concrete thread of analysis we identify, categorize and discuss all the key aspect that affect or characterize autonomic architecture stability, from theoretic analysis and network design point of view, to practical implementations and runtime solutions. Moving one step further, we propose concrete methodologies and highlight corresponding theoretic tool for addressing and studying stability problems in autonomic networks [15].

Acknowledgement

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

References

1. ITU: The FCAPS management framework, ITU-T Recommendation M. 3400, Telecommunications management network, Copyright ITU (2001)
2. Chaparadza, R., Papavassiliou, S., Kastrinogiannis, T., Vigoureux, M., Dotaro, E., Davy, A., Quinn, K., Wodczak, M., Toth, A.: Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering. In: Future Internet Assembly (FIA) book in Europe, pp. 136–147. IOS Press, Amsterdam (2009)
3. Diao, Y., Hellerstein, J.L., Parekh, S., Griffith, R., Kaiser, G.E., Phung, D.: A control theory foundation for self-managing computing systems. IEEE Journal on Selected Areas in Communications 23(12), 2213–2222 (2005)
4. Jiang, T., Baras, J.S.: Fundamental Tradeoffs and Constrained Coalitional Games in Autonomic Wireless Networks. In: Proc. of 5th Int. Symp. on Modeling and Opt. in Mobile, Ad Hoc and Wireless Networks, WiOpt 2007, pp. 1–8 (2007)
5. Boutaba, R., Xiao, J., Zhang, Q.: Toward Autonomic Networks: Knowledge Management and Self-Stabilization, Autonomic Computing and Networking, pp. 239–260. Springer, Heidelberg (2009)
6. Greenberg, A., Hjalmtysson, G., Maltz, D.A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhan, J., Zhang, H.: A clean slate 4D approach to network control and management. ACM SIGCOMM Comp. Com. Review 35(5), 41–54 (2005)
7. Ballani, H., Francis, P.: CONMan: A Step Towards Network Manageability. ACM SIGCOMM Computer Com.Review 37(4), 205–216 (2007)
8. Fu, F., Van der Schaar, M.: A New Systematic Framework for Autonomous Cross-Layer Optimization. IEEE Trans. on Veh. Tech. 58(4), 1887–1903 (2009)
9. Rad, H.M., Huang, J., Chiang, M., Wong, V.: Utility-optimal random access: Reduced complexity, fast convergence, and robust performance. IEEE Tran. on Wireless Com. 8(2), 898–911 (2009)
10. Felegyhazi, M., Hubaux, J.P.: Game Theory in Wireless Networks: A Tutorial. In: Proc. of Infocom 2006, Barcelona, Spain (April 2006)
11. Saad, W., Han, Z., Debbah, M., Hjorungnes, A., Basar, T.: Coalitional Game Theory for Communication Networks: A Tutorial. IEEE Signal Processing Magazine, Special Issue on Game Theory 26(5), 77–97 (2009)
12. Mortier, R., Kiciman, E.: Autonomic Network Management: Some Pragmatic Considerations. In: Proc. of 2006 ACM SIGCOMM, NY, USA, pp. 89–93 (2006)
13. Prakash, A., Chaparadza, R., Theiz, Z.: Requirements of a Model-Driven Methodology and Tool-Chain for the Design and Verification of Hierarchical Controllers of an Autonomic Network. In: First International Conference on Models and Ontology-based Design of Protocols, MOPAS 2010 (June 2010) (to appear)
14. Tcholtchev, N., Chaparadza, R., Prakash, A.: Addressing Stability of Control-Loops in the context of the GANA architecture: Synchronization of Actions and Policies. In: Intl. Workshop on Self-Organizing Sys., IWSOS, Zurich (2009)
15. EC funded- FP7-EFIPSANS Project, http://efipsans.org/