Autonomics and SDN for Self-Organizing Networks

Giorgos Poulios, Kostas Tsagkaris, Panagiotis Demestichas, Abdoulaye Tall, Zwi Altman, Christian Destré

Autonomics and SDN for Self-Organizing Networks

Giorgos Poulios, Kostas Tsagkaris, Panagiotis Demestichas
University of Piraeus Research Center, Piraeus, Greece
{gpoulios, ktsagk, pdemest}@unipi.gr

Abdoulaye Tall, Zwi Altman, Christian Destré
Orange Labs, Issy les Moulineaux, France
{abdoulaye.tall, zwi.altman, christian.destre}@orange.com

***Abstract*— This paper presents the study of the relationship between Autonomic Network Management (ANM) and Software-Defined Networking (SDN) under the prism of Long Term Evolution (LTE) Self-Organizing Networks (SONs). The ANM and SDN paradigms, besides sharing a few common goals, have shown to be complementary to each other in terms of level of abstractions and expectations. In particular, we consider the Unified Management Framework (UMF) introduced in the FP7 UNIVERSELF project, which focuses on higher level self-* functionality, often assuming as given a fictional adaptation layer between such autonomics and the managed infrastructure. On the other hand, SDN architectures provide by design a uniform control substrate for the programmatic management of network resources through vendor-agnostic APIs. However, despite the popularity of flow-based control in core networks and data centers, a similar widely-adopted abstraction to that of the "flow" have yet to be defined for radio access and SONs. In this work we propose such a novel abstraction layer designed to realize SON programmability. The prototype Autonomic SDN (AutoSDN) controller has been integrated with UMF to self-optimize an LTE-Advanced heterogeneous network, enabling SON functions to be provided by 3rd parties and to be *hot-plugged* into the network.**

Keywords—autonomic; software defined; SDN; self-organizing; SON; UMF; wireless; radio

I. INTRODUCTION

Modern radio access networks (RANs) have shown to exhibit high management and operation complexity. This is an effect of the heterogeneity among the deployed technologies and in most cases attributed to their lack of interoperability. Automation of the otherwise repeated, time-consuming and error-prone human configuration is just one of the reasons why Autonomic Network Management (ANM) and Self-Organizing Networks (SONs) are promising approaches in the field of radio access. A Typical example of the need for self management is the massive small cell deployment, which, while envisaged to cover future capacity and coverage needs, is expected to only increase the operation and maintenance expenditures. Another reason contributing to their adoption is the high responsiveness and computational capacity of (either software or hardware) autonomic mechanisms compared to that of human experts.

In the context of ANM, the Unified Management Framework (UMF) has been proposed by the EU FP7 project UNIVERSELF in an attempt to tackle exactly this heterogeneity by means of self-management [1]. UMF has taken the concept of SON further to study how self-* functions -called Network Empowerment Mechanisms (NEMs) in UMF terminology- can be incorporated and managed

978-1-4799-5863-4/14/$31.00 ©2014 IEEE

in a running ANM ecosystem *on-the-fly*. The UNIVERSELF approach to that was to provide specification of, among others, a common management interface for all compliant autonomic functions regardless of their nature or purpose. On the contrary though, interfacing between the functions and the respective managed entities (i.e. the network elements) is not specified by UMF, despite the fact that a uniform adaptation layer to the underlying protocols/interfaces is deemed necessary.

Evidently, SONs are some of the most interesting use cases for such ANM framework as they inherently rely on self-* control loops. However, the current SON landscape has functions be either provided by equipment vendors (*vendor-domain* SONs) or as part of the NMS/OSS (*operator-domain* SONs). The former, may be provided as distributed control loops with local network view and high responsiveness, or as part of the vendor-provided, closed-source and proprietary Element Management Systems (EMS). Due to the black box approach followed and the lack of standardized, open interfaces, such solutions often trap operators in situations known as *vendor lock-ins*, in which they are forced to extend/upgrade their infrastructure from the same manufacturer. We claim that two significant properties are missing from vendor-domain SONs: a) "*openness*" towards the owner of the equipment for the sake of development/ customization, and b) universal interfaces for the sake of management and smooth integration with Network Management Systems (NMSs).

Such phenomena are avoided in the case of operator-domain SON functions, i.e. implemented as part of the NMS/OSS (Operations Support System). Although the latter can also provide with global network view, SON functions at this level usually lack responsiveness and are restricted to operate at larger time-scales. Moreover, due to the lack of interoperability between EMSs and the NMS, substantial effort is required for their integration.

Inspired by the well known example of OpenFlow, in this work we show how a software-defined SON can make the best out of the two worlds by pinpointing the appropriate logical abstractions and following the SDN principles. However, although the flow-based switching of packets may seem promising in core networks, we claim that the cornerstone of a SON is instead, the *function* coupled with the *Key Performance Indicator* (KPI), and the *configuration parameter*. Built upon these abstractions, the Autonomic SDN (AutoSDN) controller forms a novel type of control substrate and provides the missing, uniform adaptation interfaces for autonomics/UMF-NEMs in an instantiation to SONs.

The rest of this paper is structured as follows: section II is an analysis of the state-of-the-art in autonomics and wireless SDN while section III presents the AutoSDN design and architecture. Section IV presents the results of the AutoSDN prototype on a 4G LTE-Advanced use case, and finally, a qualitative analysis and conclusions are given in section 0.

II. Related Work

A. Autonomics

Several architectures have been proposed in the literature to realize the ANM paradigm. Starting from the widely recognized manifest of IBM for autonomic computing [2], the introduction of autonomics into the areas of network management [3][4] has been the goal of several endeavors in the research community. The well-known “Monitor Analyze Plan & Execute – Knowledge (MAPE-K)” loop in [2] has also been adopted as part of the lifecycle of autonomic, Network Empowerment Mechanisms (NEMs) of the UMF [1]. According to that, an autonomic system needs to have capabilities to monitor its environment, be aware of at least static information (a.k.a. profiles), learn from its own observations (knowledge) and make decisions respecting a set of policies [5]. Many impactful research projects that also influenced standardization bodies [6][7][8][9] were built upon these principles.

One of these standardization bodies, the ETSI AFI group, is standardizing an evolvable holistic architectural reference model called GANA (Generic Autonomic Networking Architecture) [9], for autonomic network engineering, cognitive networking and self-management. The corresponding self-management architecture of UMF as proposed in the European project UNIVERSELF is semi-distributed with autonomic control loops scattered across the network and a logically centralized core responsible for their governance (GOV), coordination (COORD) and knowledge exchange (KNOW) [1]. In that, the control loops (i.e. the NEMs), besides following the MAPE-K loop, are modeled along

3 main axes: their *objective*, *method* and *context*. The resulting information model has been developed as an extension to the Shared Information/Data (SID) [10] framework of the TeleManagement Forum (TMF). In section III we present how the UMF can be mapped to the proposed SDN architecture for SONs.

B. *Software Defined Networking in wireless networks*

Although OpenFlow [11] has been designed for experimentation in campus networks, it has gained increased recognition in the last years with applications mostly on core networks and data centers. OpenRoads [12] however is an extension to OpenFlow for experimentation in mobile networks focusing on mobility issues. OpenRadio [13] approaches wireless SDN through a software abstraction layer that enables wireless protocol programmability at the physical and MAC layers. From a different perspective, SoftCell [14] attempts to support fine-grained policies, in a flexible and scalable architecture by placing a set of middleboxes in the cellular core. SoftRAN [15] on the other hand, approaches the software-defined RANs through the concept of the so-called "Big Base Station" which abstracts out all the base stations deployed in a geographical area into a virtual one. Based on that, and similar to the traditional SDN architectures, SoftRAN proposes a logically centralized controller providing global network view to applications.

III. AutoSDN Design And Architecture

In order to provide global network view, the AutoSDN architecture follows the pattern of the logically centralized controller too. Since we focus on SONs though, the main abstractions defined are these of the *metric* (which includes the *KPI*), the *configuration parameter* and the *SON function*. The controller is therefore designed to be an execution environment for plug'n'play functions, i.e. handle and provide them with (potentially filtered or aggregate) metric data, evaluate, and apply their output-parameter configurations to network elements. The overall architecture, is depicted in

Fig. 1. At NMS/OSS-level, the UMF core gives the operator total control over the SON functions that are implemented as UMF NEMs, e.g. to instantiate/delete/monitor/start/stop them, assign managed elements, apply policies or even set their pace of execution.

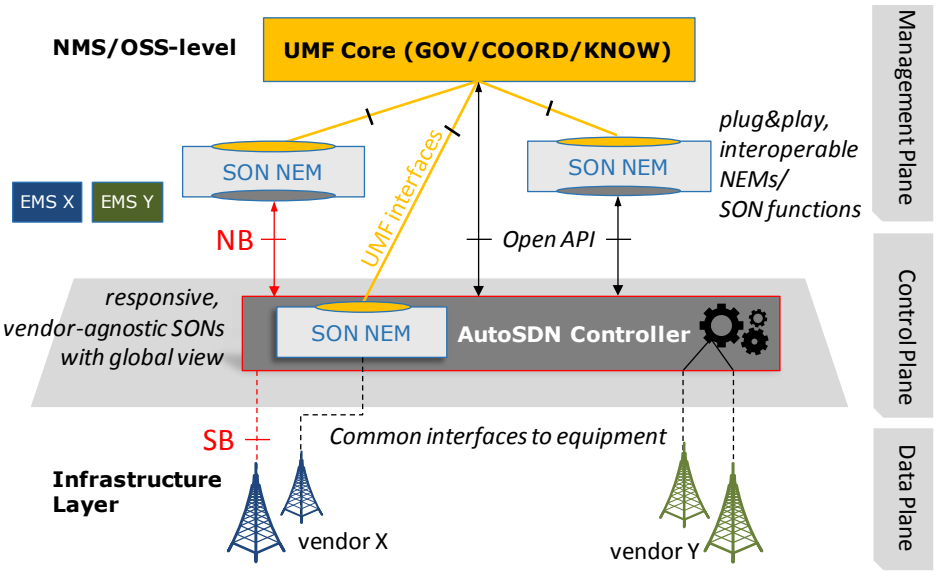


Fig. 1. High level AutoSDN architecture

NEMs, in turn (as well as the UMF Core), through the controller's northbound (NB) API can:

- discover existing network elements and topology,
- retrieve or configure the value of parameters, and
- monitor KPIs or subscribe to events related to KPI value changes.

On the other end, the southbound (SB) interface is also an API, unlike most other SDN approaches where it is a protocol. Through that, network elements, metrics, parameters and their associations can be introduced to the system. Metric and parameter values can then be updated though the SB periodically, while the controller undertakes the synchronization with any interested northbound entities.

The NB, SB and UMF interfaces are completely vendor-agnostic. The controller in particular, makes no assumption on the types[1] of underlying network elements, their parameters or metrics. The static structure of, for instance, the Management Information Base (MIB) of SNMP was intentionally avoided to ensure broader applicability and potential extensibility to new (even, types of) metrics or elements. As analyzed in the following paragraphs, instead of organizing a given set of parameters in large information bases, AutoSDN models and handles them in an abstract way.

Implementation-wise, in order to enhance remote access, AutoSDN SB and NB interfaces are also exposed as HTTP-RESTful web services. The data formats supported by these services are XML, JSON and plain text (where possible) while the choice is left up to the client-side developer.

A. Metric, Parameter and Element Model

Metric in AutoSDN is considered as anything measurable, orderable, read-only, and implementation-wise, of numeric/double data type. It is defined as the triplet *specification*, *context* (where it applies) and *source* (where to get its value from). The multiplicity between them, in the scope of the overall system, is one specification to many source-context pairs. That is because the *metric specification* is a static, global and unique description of the type of a metric. Among others, it includes a name, a measurement unit, a minimum, maximum and a default value. The *metric context*, often denotes the network element which the metric concerns, but it might also be some abstract concept such as a subnet, or a cell cluster. Finally, the *metric source*, is the endpoint for acquiring metric values, and is not strictly required to be the producer of the data, but rather the logical source in the scope of AutoSDN. The rationale behind it is to expose a common data-retrieval interface regardless of the provider or the type of metric. Regarding its positioning, a metric source may reside either at the infrastructure layer or inside the controller, enabling both centralized and distributed data dissemination schemes. In both cases, it is registered in the controller with one of the metric specifications so that interested entities may discover it.

Metric sources also generate various kinds of events regarding the state of their value with respect to a range, a time interval or the previous value. Fig. 2 depicts the possible ways a SON NEM enabled by such events, may retrieve metric data.

[1] The terminology adopted in AutoSDN for "type" and "instance" of a concept x, is "x specification" and "x" respectively (where x can be an element, metric, or parameter), similar to UMF, and influenced by [10].

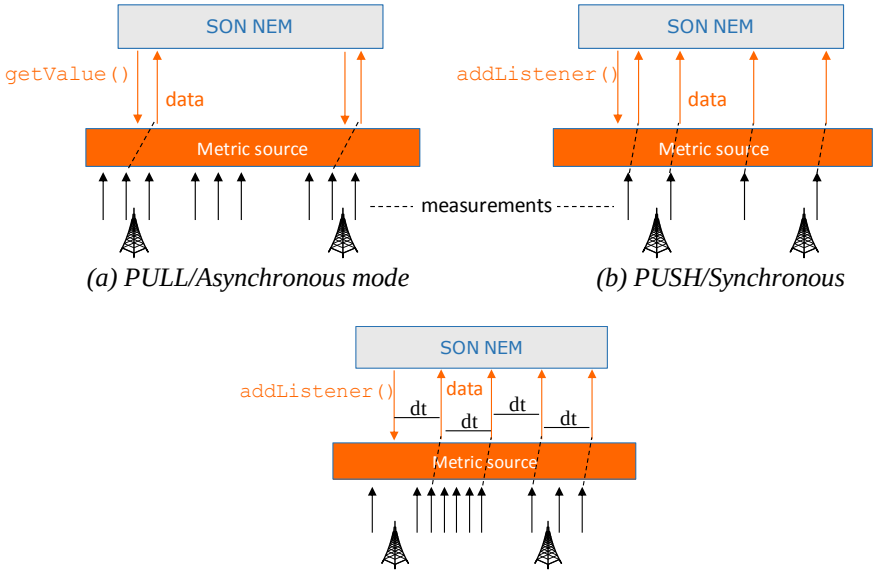


Fig. 2. The three supported modes of metric data exchange by AutoSDN

In (a), the NEM simply invokes the respective method in a request-reply fashion, hence denoted as *PULL*, and *asynchronous* because the value held by the NEM is not necessarily the most up-to-date one. On the contrary, in (b), the NEM subscribes as a listener to value-change events. Upon change, the updated value gets *pushed* to it by the source, hence the two remain *synchronized*. Lastly, a time-based type of event is used in (c), which periodically notifies the listener of the (then) latest value of the metric.

Parameters in AutoSDN also have a *specification*, an *adapter*, and an associated *network element*. Similarly to metrics, the *specification* corresponds to the type of the parameter specifying its name, data type, access type and unit of measurement, if any. The *adapter*, in analogy to the metric source, provides a common interface to read or write the parameter value. Respectively, the multiplicity between them is one specification to many adapter-element pairs.

Elements in turn, are a conglomeration of parameters and other elements. This recursive definition allows for reusability of "types"/descriptions of equipment at finer granularities. For instance, the antenna or Remote Radio Head (RRH) of a base station (BS) can be considered as elements of the greater BS-element. This way, BSs with different specifications may share the same type of antenna or RRH.

B. SON Function Model

The three most important parts of a function definition are its **inputs**, **output** and **body/code**. However, besides the types of inputs/output (resp. the metrics or parameters), it is required to specify *which particular instances of metrics and parameters* (in other words, *which elements*) to apply the function to, along with *when* to evaluate and apply it. For the former, AutoSDN defines a list of tuples accompanying any function called **invocations**. Each of them specifies the elements/context of the parameters/metrics to use as inputs and output. For the latter, the concept of **trigger** is defined, based on either timers or metric event changes. Any function is again accompanied by a set of triggers specifying when it should be evaluated and applied.

All the aforementioned properties are encapsulated by a `SONFunction` data structure and can be passed to the controller through the NB API for installation and execution during runtime. Alternatively, they may be implemented-by/embedded-in the SON NEM, in which case the NB would be used only for getting/setting the required/computed values.

C. *Components and Functional View*

Fig. 3 depicts a functional overview of the controller. From left-to-right, the **metric collector** is the endpoint for pushing data to metric sources, in case the latter are maintained by the controller. Capability to push bulk data has been provisioned to minimize the network footprint by combining many web service invocations into a bulk one. Regardless of sources being maintained by the controller or the infrastructure layer, the **metric registry** keeps track of all metric specifications and sources for discovery from the NB. The **configuration manager** in turn, keeps track of the parameters, provides an interface to their adapters and ensures their access restrictions are respected by the NB entities.

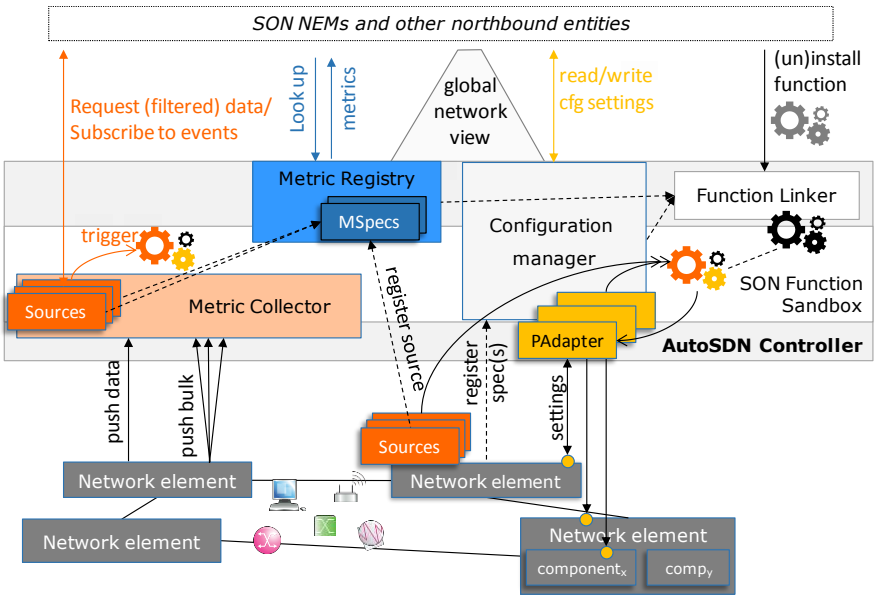


Fig. 3. Functional overview of the AutoSDN controller

The **function linker** undertakes compilation and linkage of `SONFunction` objects to the specified inputs, outputs and triggers. Linked functions are then forwarded to the **SON function sandbox** for execution. The sandbox is required because functions are coded in Javascript and are unknown a-priori, while executed within the controller's process space, the latter in Java. Hence, the sandbox is in practice a Javascript engine, namely Rhino [16], with restricted IO access to other functions, the controller internals or the host operating system.

IV. Instantiation to a 4G LTE HetNet

A. *Theoretical ground of the considered SON functions*

According to 4G LTE-Advanced, small cells (a.k.a. pico cells) can be used to offload macro cell traffic and to considerably increase the network capacity. This process is often denoted as load balancing, and is achieved by moving traffic from the macro cell to the small cells. Small cells can be deployed close to the cell edge where typically more macro cell resources are needed to serve UEs, and/or at hot spots. To enhance the off-loading capability, 3GPP has introduced two jointly operating mechanisms [17]: the first being the Cell Range Extension (CRE) which aims at extending the coverage of small cells by increasing the Cell Individual Offset (CIO) parameter. A UE u then chooses the best serving station according to (1) where P_s is the received pilot power for cell s, h_s^u is the pathloss from BS s to user u and CIO_s its CIO.

$$s^* = argmax_s CIO_s h_s^u P_s \tag{1}$$

However, the traffic at the extended coverage zone will suffer from interference coming from the macro cell. Therefore a second mechanism is introduced, namely an interference mitigation which, on certain subframes, mutes almost all macro BS transmissions. These subframes are denoted as Almost Blank Subframes (ABS).

An optimal operation of these two mechanisms requires SON functions to optimize the parameters and to dynamically track the traffic variations. Distributed SON functions can be modeled as local control loops of the following for

$$x_{k+1} = x_k + \epsilon g(x_k) \tag{2}$$

where x_k is the parameter of the SON function at iteration k, ϵ is a step size parameter which is typically small, to ensure the convergence of the algorithm in a stochastic environment, and the function $g(\cdot)$ is a function of one or several KPIs, and provides the direction of the update. From stochastic approximation theory [18], it is known that the mean behavior of (2) is given by the following Ordinary Differential Equation (ODE):

$$\dot{x} = g(x). \tag{3}$$

According to [18] we have the following two propositions:

Proposition 1. Let $f(\cdot)$ be a Lipschitz-continuous function, and $\bar{f}$ – a constant value. If $g(x)$ in (3) is defined as $g(x) = f(x) - \bar{f}$, then x in algorithm (2) converges to x^* such that $f(x^*) = \bar{f}$.

Proposition 2. Let $f(\cdot)$ be a concave function. If $g(x) = \nabla f(x)$ then x converges to x^* such that $f(x^*) = \max_x f(x)$.

The CIO is optimized using a Load Balancing (LB) SON between a macro cell and its neighboring small cells. It adapts the CIO_s at time step *k* as a function of the difference between the macro cell load ρ_m and the small cell load ρ_s (4)**Erreur ! Source du renvoi introuvable.**

$$CIO_s(k+1) = CIO_s(k) + \epsilon_1\big(\rho_m(k) - \rho_s(k)\big) \tag{4}$$

From Proposition 1, the LB SON will converge to a CIO for which the macro load and the small cell load are equal. A detailed proof for this algorithm can be found in [19].

The ABS SON algorithm aims at minimizing the maximum cell load by adjusting the ratio θ of muted subframes for the macro BS m, and is given by (5), with *1…p* being the indices of the small cells offloading macro cell *m*.

$$\theta_m(k+1) = \theta_m(k) - \epsilon_2 \frac{\partial\big(max(\rho_m,\rho_1,\dots,\rho_p)\big)}{\partial\theta_m} \tag{5}$$

The maximum is taken over the load of the macro cell and those of the *p* small cells in its coverage area. The loads of the cells are convex in θ_m. From Proposition 2, the algorithm (5) converges towards a mute ratio that minimizes the maximum of the loads. The convexity of the loads is shown as follows. From queuing theory [20] the loads can be written as:

$$\begin{aligned} \rho_m &= \int_{A_m} \frac{\lambda(r)\mathrm{E}(\sigma)}{(1-\theta_m)\,R_m(r)}\,dr \\ \rho_p &= \int_{CRE_p} \frac{\lambda(r)\,E(\sigma)}{\theta_m R_p(r)}\,dr + \int_{A_p - CRE_p} \frac{\lambda(r)E(\sigma)}{(1-\theta_m)R_p(r)}\,dr \end{aligned} \tag{6}$$

where $\lambda(r)$ is the arrival rate at position r, A_s is the area of cell s, σ is the download file size, $R_s(r)$ is the peak data rate offered by BS s at position r, and CRE_p is the CRE area of small cell p. These loads are convex in $\theta_m \in [\tau, 1-\tau]$ where τ is a small positive constant. Since the max function preserves convexity, their maximum is also convex. It is noted that other algorithms for ABS SON can be implemented [21].

The question of the stability of the two SON functions operating simultaneously may arise. The justification for stability is given by basic control theory results using the framework described in [22].

B. Instantiation to the proposed architecture

For the purpose of evaluation, AutoSDN and the two above mentioned mechanisms have been instantiated over a Matlab-simulated heterogeneous radio environment with macro and small cells. The macro and small cells, along with the associated metrics/KPIs, are registered to the controller through the SB API. The LB and ABS SONs -implemented as NEMs- are instantiated through the UMF dashboard and deployed over the small and macro cells respectively, during runtime. This deployment process corresponds to the installation and execution of the respective SON functions into the controller through its NB API.

Fig. 4 depicts the system setup of the prototype using 5 machines, each of them hosting a different component. Although they may all be hosted by a single machine, this setup illustrates the distribution possibilities. No particular type of connectivity is assumed between the components, other than HTTP access to the servers (through firewalls, NATs etc).

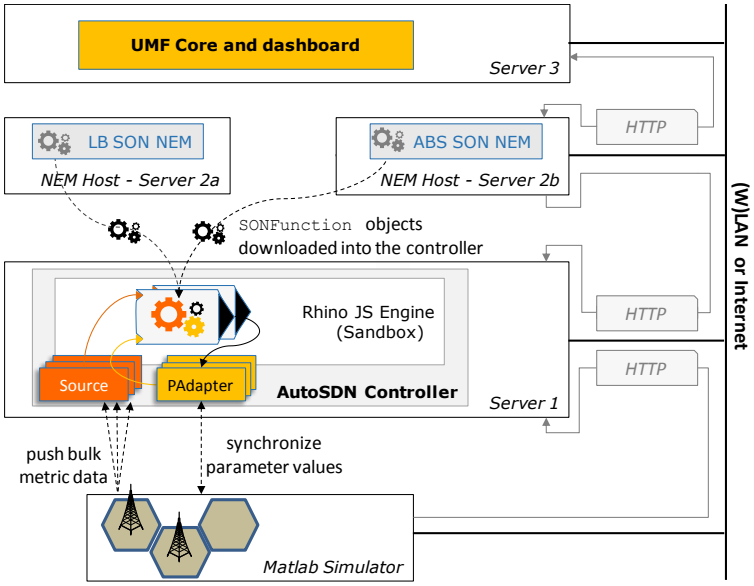


Fig. 4. System view of the AutoSDN prototype setup

The NEM hosts are servers capable of spawning new NEM processes (a.k.a. instances) upon instruction by the UMF dashboard. Although not explicitly shown in Fig. 4 for the sake of clarity, the UMF core also interacts with the controller NB to retrieve various information on the elements such as their topology and type. From the visualized network, the operator may then choose to deploy any NEM over (a set of) element(s), according to its declared types of manageable elements. In this example, upon deployment, each NEM configures and downloads its corresponding `SONFunction` object to the controller for execution. Notably, and as with all other components, servers 2a and 2b could very well be combined in a single server capable to instantiate any kind of NEM, as is common in other UMF instantiations.

Fig. 5 depicts the overall simulated network as shown in the UMF dashboard (and retrieved from the controller) with 21 macro and 12 pico base stations.

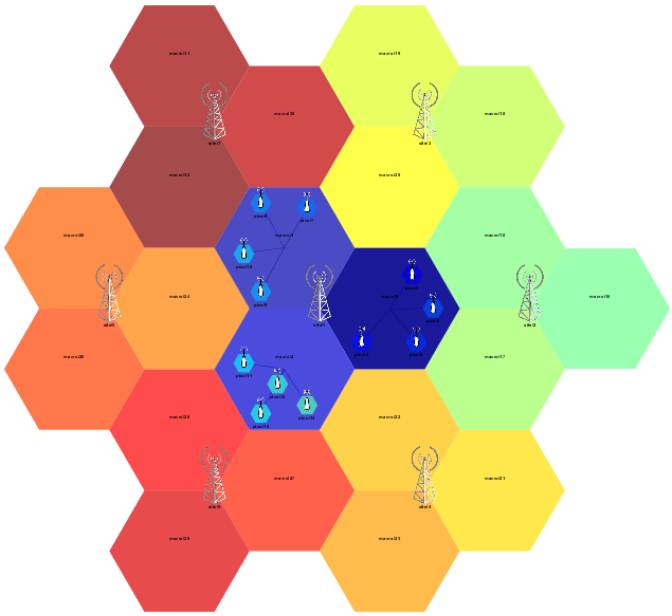
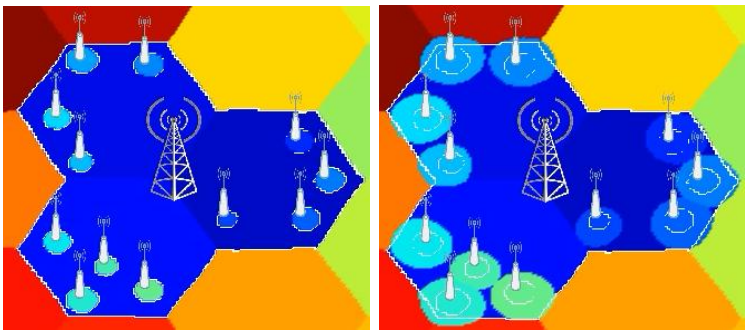


Fig. 5. The network view of the UMF dashboard, as discovered through the AutoSDN controller, and simulated by Matlab.

C. *Results*

Only the three central sectors along with their small cells are included in the results. The surrounding cells (in shades of red and yellow) have been placed to simulate interference. (a) (b)

Fig. 6 depicts two snapshots of the serving-cell map, prior and after operation of the LB SON (using $\epsilon_1 = 0.05$ and $\epsilon_2 = 5 \times 10^{-4}$). At the initial state of the network, around 80% of the traffic is served by the macros, hence, as expected, the LB SON increases the CIO of the picos.



(a) (b)

Fig. 6. (*left*) coverage map produced by the simulator with no SON functions; (*right*) increased CRE of the picos after activation of the LB SON

Fig. 7 depicts the resulting evolution of the CIO (*top-left*) and ABS ratio (*top-right*) due to the LB and ABS SONs respectively, with the latter adjusting itself following the former, autonomously. As a result, the mean number of users in the system is reduced (*bottom-left*), due to higher data rates which result in lower service times, while as illustrated indicatively for macro/0 (*bottom-right*), a portion of the macro traffic is off-loaded to their picos, freeing up precious macro BS resources.

Performance-wise, the simulation was running in around 120 times faster than real-time. The controller was run by a 2.4Ghz CPU core and communicating over WLAN with the simulator. However, the bottleneck was still the simulation. By using bulk data transfers and well-packed JSON objects (yet no compression), only a small portion of the time was spent interacting with the controller. This observation indicates that in a real life pace, the proposed system will be efficient.

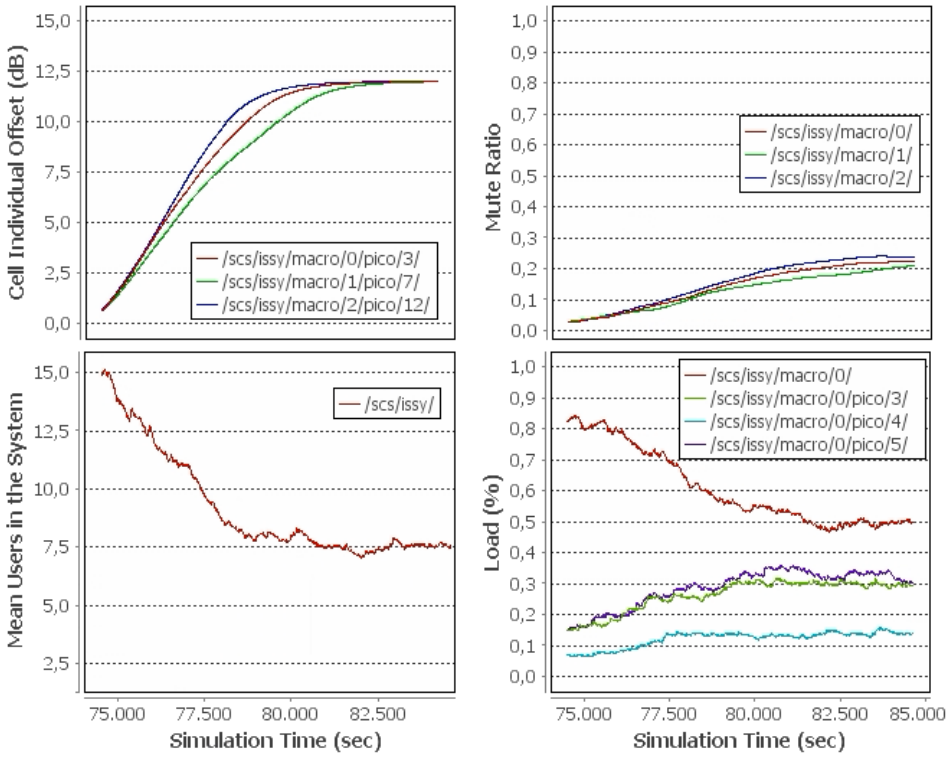


Fig. 7. Evolution of the KPIs and the key SON parameters, as visualized by the AutoSDN controller

V. Qualitative Analysis and Conclusion

Over the current SON situation, AutoSDN has considerable qualitative benefits. *Vendor lock-ins* are tackled by the separation of data and control planes, and in particular, by shifting control plane functionality towards the operator domain. However, two basic assumptions made in this case are that a) manufacturers agree to "open" their equipment and comply to the AutoSDN SB, and b) that the controller is implemented in an efficient way. Especially regarding the former, we claim that, since "software-ization" is the modern trend of the networking world, any vendor supporting open standards increases its competitiveness. Should such preconditions be satisfied, the controller can then provide an intuitive, uniform and vendor-agnostic NB interface to developers. Further to that, cross-vendor and cross-segment SON functions may be implemented, which is something rather unusual in current SONs except for OSS-level functions.

In turn, given such a universal NB API together coupled with plug'n'play capabilities, SON function development may well be outsourced to 3rd parties, forming a SON market or open source communities, thus encouraging competition, innovation and experimentation. SON function development-time and time-to-deployment is also greatly reduced as the peculiarities of the underlying technologies are abstracted.

Furthermore, one significant benefit of the AutoSDN approach in order to compensate for the rapid evolution of RAN technologies, is that new metrics, parameters, elements, and SON functions may be introduced to the system without re-compiling or restarting it. This is particularly important to reduce efforts required for upgrade and integration of new infrastructures by operators.

ACKNOWLEDGMENT

This work is part of the research project AutoSDN involving the University of Piraeus Research Center and Orange, under contract no. D02191.

REFERENCES

[1] UNIVERSELF project Deliverable 2.4 “UMF Specifications - r3”, available at http://goo.gl/5O4TuK, 2013

[2] J. O. Kephart, and D. M. Chess. "The vision of autonomic computing." *Computer* 36.1 (2003): 41-50.

[3] Autonomic Network Management Principles: From Concepts to Applications, Edited by N Agoulmine, ISBN 978-0-12-382190-4, *Elsevier Academic Press Publications.* 2011

[4] Z. Movahedi, et al. "A survey of autonomic network architectures and evaluation criteria." *Communications Surveys & Tutorials*, IEEE 14.2 (2012): 464-490.

[5] P. Demestichas. "Introducing cognitive systems in the wireless B3G world: Motivations and basic engineering challenges." *Telematics and Informatics* 27.3 (2010): 256-268.

[6] R. Chaparadza, et al. "Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering." *Future Internet Assembly*. 2009.

[7] ETSI, TC RRS, “Reconfigurable Radio Systems (RRS); Functional Architecture (FA) for the Management and the Control of Reconfigurable Radio Systems”, TR 102 682 v1.1.1, July 2009

[8] ETSI GS AFI 001: Autonomic network engineering for the self-managing Future Internet (AFI): Scenarios, Use Cases, and Requirements for Autonomic/Self-Managing Future Internet. June 2011

[9] ETSI GS AFI 002: Autonomic network engineering for the self-managing Future Internet (AFI): GANA Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management. Aug-Sep 2011

[10] Information Framework (SID) reference page, http://goo.gl/o03Xxq, TeleManagement Forum, July 2013

[11] N. McKeown, et al. "OpenFlow: enabling innovation in campus networks." *ACM SIGCOMM Computer Communication Review* 38.2 (2008): 69-74.

[12] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, “Openroads: empowering research in mobile networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 125–126, January 2010

[13] M. Bansal, et al, “OpenRadio: A programmable wireless dataplane,” in *Hot Topics in Software Defined Networks*, pp. 109–114, 2012

[14] X. Jin, L. E. Li, L. Vanbever, and J. Rexford. "SoftCell: Taking Control of Cellular Core Networks". In TR-950-13, Princeton University, 2013

[15] A. Gudipati, et al. "SoftRAN: Software defined radio access network." *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013.

[16] Mozilla Rhino, http://goo.gl/3nXi6N, accessed at June 2014

[17] “Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2”, 3GPP, TS 36.300 v10.11.0, Sep. 2013

[18] V. S. Borkar, Stochastic Approximation: A Dynamical Systems Viewpoint. Cambridge University Press, 2008.

[19] R. Combes, et al. “Self-organization in wireless networks: a flow-level perspective,” in Proceedings of IEEE INFOCOM, 2012.

[20] T. Bonald and A. Proutière, “Wireless downlink data channels: User performance and cell dimensioning,” in ACM Mobicom, 2003

[21] A. Tall, Z. Altman, and E. Altman, “Self organizing strategies of enhanced ICIC (eICIC)”, in Proceeding of the 12th Intl. Symp. on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt 2014, Hammamet, Tunisia, May 2014.

[22] A. Tall, R. Combes, Z. Altman, and E. Altman, “Distributed coordination of self-organizing mechanisms in communication networks,” to appear in IEEE Trans. on Control of Network Systems (TCNS), 2014.