# Using an Identity Plane for Adapting Network Behavior to User and Service Requirements

Pedro Martinez-Julia(✉) and Antonio F. Skarmeta

Department of Communication and Information Engineering,
University of Murcia, 30100 Murcia, Spain
{pedromj,skarmeta}@um.es

**Abstract.** The advent of Software Defined Networking (SDN) has opened the door to new network functions that were difficult or even impossible to have. This has been the case of typically complex network management operations, which now can be layered on top of SDN controllers in order to adapt network behavior to achieve some objectives or quickly react to network events so network consistence is unaltered by them. However, users and services have little to say in current SDN architectures. In this paper we discuss how to use an *Identity Plane* to carry user and service identities and requirements to network controllers, which would contact a management service that follows the management model proposed by Autonomic Computing (AC) to know the necessary changes to adapt the network behavior to such requirements.

**Keywords:** Future internet · Identity · Overlay network · Autonomic computing · Self-management · SOA

## 1 Introduction

In current networks, most network management operations are performed using certain mechanisms and protocols for monitoring and configuring network elements, from virtual to physical elements and from hosts to network equipments. Even though those mechanisms are normally used by specific applications that permits administrators to manage multiple elements from a central place, all tasks usually need human intervention. This behavior is also spread along the current Internet but, as the number of network elements grows, this task is becoming more and more complicated to accomplish. Therefore, a new management paradigm is emerging from the Autonomic Computing (AC) initiative. It will overcome future requirements on self-management of systems and services, which are key challenges for the Future Internet (FI) [1,17,19].

Apart from resolving the issue with the rapid growth of systems to manage, the AC also address the added problem found in the also rapidly growing computer systems complexity, dynamism, and heterogeneity. Thus, AC systems are defined as "computing systems that can manage themselves given high-level objectives from administrators" [12]. This definition encompasses the key

R. Agüero et al. (Eds.): MONAMI 2015, LNICST 158, pp. 253–265, 2015.
DOI: 10.1007/978-3-319-26925-2_19

principle behind AC: Administrators (humans) set the rules (policies) by which systems should be guided and those systems are responsible of enforcing them. This way, AC presents a good solution to add self-management capabilities to modern networks and services, and so is supported by the Future Internet Assembly [7] on its MANA position paper [8] on which autonomic network management plays a fundamental role, incorporating to the FI service model the main activities found in AC.

On the other hand, the Software Defined Networking (SDN) model is experiencing a huge growth, providing the necessary underlying mechanisms to implement control and management operations with little or no impact to underlying elements. Such model clearly separates control and data planes, which is an important feature, but also provides the necessary interfaces to build network services and applications on top of network controllers, which means the breaking of network ossification and the provisioning of huge flexibility to the network.

This has led us to define a mechanism that connects an *Identity Plane* [15] to the SDN controller and the necessary connection points for them to properly address management operations and reflect their results into the network. Intermediate network elements will not have to be changed because the connection is performed through the SDN control plane. This way, network entities will be able to declare their network requirements and the network will be able to respond to such declaration by taking the appropriate determinations to meet those requirements as best as possible. All of this will be done without human intervention but some humans, particularly the network administrators, have to define the policies to which management and control operations will be enforced to accomplish.

The remainder of this paper is organized as follows. First, we introduce the motivation of the present work and the challenges it exposes in Sect. 2. Then, in Sect. 3 we describe the proposed solution, composed of network management modules and the interconnection to the *Identity Plane* and the SDN control plane. In Sect. 4 we present an experimental instance of the proposed solution and discuss the experimentation results we have obtained with it. In Sect. 5 we briefly analyze the related work and, finally, in Sect. 6 we conclude the paper and introduce some hints for the future.

## 2 Future Identity-Based Network Management

The search towards future networks has exposed many challenges [11] that have promoted the design of new architectures that deprecate the current network models. Our vision is that many of them will coexist in the future so it is desirable to combine their qualities to provide the best service to network users. On the other hand, the advances in technology are merging our real lives with our digital lives, so user identities and contexts must not be only considered at application level but also at network level, allowing the establishment of zones of privacy (as in real life), as well as controlled identity linkability and information disclosure.

These challenges have motivated our work in a new identity-based network architecture [13–15] that builds an *Identity Plane* that interacts with users to know their intentions and willingness to use multiple devices in a communication, which requires more mechanisms than just network mobility. As user identities are delicate, every decision is taken with a maximal constraint: privacy and data protection. It leads us to avoid the disclosure of the relation of data and identities, the identification via IP or MAC address, and the ability to keep the privacy across layers, cross-layer security. Thus, it is mandatory to use identities to address communication parties, instead of using identifiers or locators.

To achieve these goals, the architecture we propose is heavily based on the overlay network concept and the possibilities offered by SDN. On the one hand, an overlay network is used to address entities by their identities, so entities themselves are the communication endpoints and they can reach each other without dealing with network location and keeping their privacy and overall security. On the other hand, the mechanisms provided by SDN are used to embed network sessions into the underlying infrastructures while ensuring they commit the necessary security and the requirements specified by communication parties.

Since this new architecture is not bound to any specific underlying network and since they can be combined during communication, the control mechanisms provided by SDN are used to establish, configure, and release communication paths among communicating entities. Such paths are defined by their communication properties or parameters that, among others, are: source and destination endpoints, represented by the rules accepted by the SDN, including the addressing scheme of the specific underlying networks; service type, or the operation that is requested to the network: send a file, ask for a file, web browsing, voice call, etc.; traffic behavior (variable/constant bit rate, rate + strength, etc.); maximum delay and throughput; and levels of priority, security, and privacy.

At the end, each underlying network will use these parameters when reserving the necessary resources to create the requested low-level communication paths. Moreover, these paths will not be static but dynamic, so they support mobility and other environment changes. However, path management must not cause a significant increase in complexity or disrupt the normal network operation. Finally, due to dynamic interactions, the complexity of the operations, and the number of elements that can be involved, the operations must not need the constant supervision of network administrators, just under enforcement of the policies they set.

These requirements have led us to opt for AC principles [12] to design the management blocks of the solution. AC can provide the required features by being stimulated by network control events and with very little disruption to end or intermediate network entities. As described in Sect. 5, current proposals for autonomic network management can not meet these requirements while being integrated with the *Identity Plane* and the SDN control. Therefore, as described in the following section, we have designed a simple management approach and included it into the integrated solution.

## 3 Proposed Solution

As introduced above, the *Identity Plane* promoted by our architecture is governed by the Domain Trusted Entity Infrastructure (DTEi) and connected to

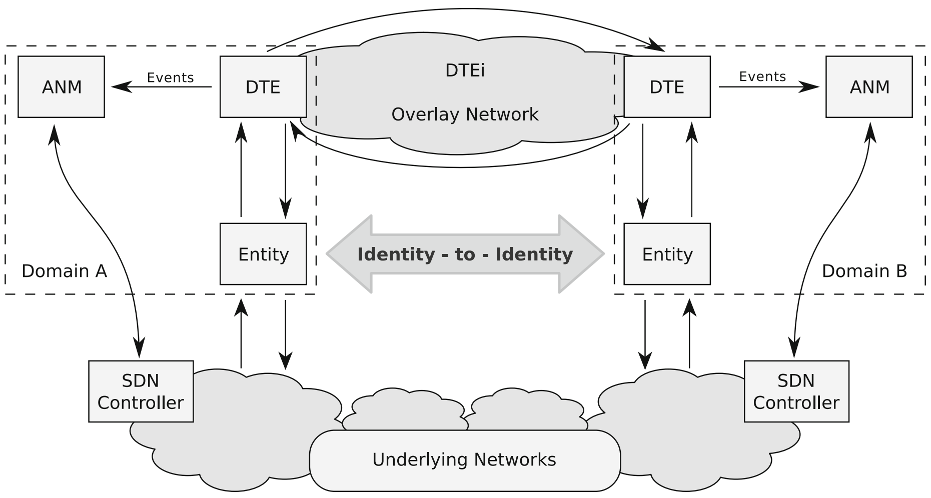

**Fig. 1.** Overview of the integrated architecture.

communicating entities in order to allow them to operate securely. This way, the DTEi is privately aware of both identities and objectives of the entities that communicate so it is in the precise position to determine the requirements of communication instances and ask the network to build the necessary paths. However, this task is out of the scope of the DTEi itself so an external component is introduced to perform it. This component is the Autonomic Network Manager (ANM), whose specific design is discussed at the end of this section.

The ANM is responsible of monitoring network operations by receiving events from the DTEi, determine which actions should be taken on response to such events, and communicate such actions to the SDN controller so it can enforce them to the network. Typical network events would be to establish new communication sessions or the movement of an entity from one network to another. In general, it would be required to change the network parameters very frequently in response of the dynamism of requirements specified by communicating entities but also in response of changes of the network. This also implies that the SDN controller will report to ANM the events related to the network paths it manages.

Instead of explicitly indicating the necessities to the ANM, it will be able to infer them by knowing what happens in the environment, like knowing when a network session is starting between two entities. To guide the whole management, the ANM will do an extensive use of policies, both to guess the reactions to certain events and to check if an operation is allowed or not. Those policies are set by network administrators, together with the necessary statements to match complex events from many simple events.

From now on we discuss how to integrate the ANM with the *Identity Plane* and thus how this solution meets the requirements commented in the previous section. Figure 1 illustrates how we propose to integrate them. It shows how network elements from different domains interact to achieve the global autonomic management objectives. The elements and their functions are described as follows:

– The DTEi, formed by the union of all its instances via an overlay network, is the main element of the *Identity Plane.* The overlay network is used to decentralize its operation across all identity/network domains. Its main function, as described above, is to mediate in communication negotiations for the entities (communication parties) of each domain. Thus, the DTEi manages, for each communication, the session establishment, the security aspects, etc. Thus, the DTEi is also in place to send the necessary events for the ANM.
– Entity represents a communication party. As commented above, entities can be persons, software, machines, things, etc. Each entity relays its network operations to its corresponding DTEi instance but the final data exchanges are performed through the data plane of the underlaying network.
– ANM is the element that watches its environment by receiving events from DTEi instances and the SDN controller. There is a different ANM deployed in each domain, connected to the DTEi instance of such domain. It receives the events, analyzes the environment, checks the policies, and decides what to do in response. Then, it will contact the SDN controller to communicate such decisions so the underlying network meets the necessary requirements.
– SDN Controller represents the controller of the current domain of the underlying network. It will report to the ANM the changes in the network regarding the communications it is managing. Also, it will receive requirements from the ANM to be enforced into the network. Those requirements are mainly represented by communication parameters, such as bandwidth, latency, security level, etc.

That said, the ANM is only coupled with the *Identity Plane* by means of the messages (events) sent by the DTEi instance of its domain, so it respects the high decoupling design principle, which is widely recommended in network architecture design. Thus, this point is the main and only point of interaction between the two architectures. As it is totally asynchronous, the network operations are not delayed or disrupted by the management operations. Finally, the inter-domain nature of management and control architectures allows the integration and interaction of different domains. Below we discuss the internals of the ANM.

## 3.1 Autonomic Network Manager

As we will analyze in Sect. 5, existing proposals for autonomic network management are centered in the interior part of the network so they do not consider entities into such task. Also, they do not use proper identification mechanisms and they are difficult to connect to current SDN control plane in a lightweight, non-intrusive manner. Therefore, in the integrated solution we have included our own approach to autonomic management.

The management solution is designed as a service-oriented architecture. Thus, it has a different service for each activity defined in AC, together with

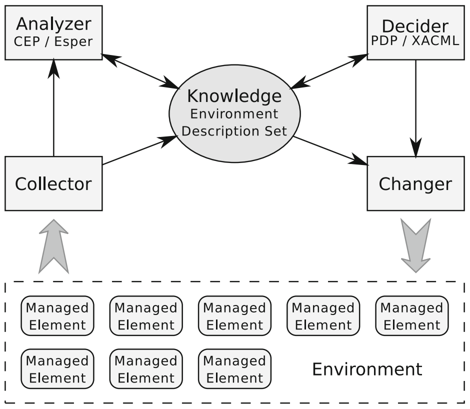

**Fig. 2.** Architecture of the management component, the ANM.

the necessary components to integrate them and help them to be integrated with other services or applications. These services are focused on genericity and flexibility. Instead of concentrated in the construction of certain solution, the services can be combined in many ways and with other services.

As depicted in Fig. 2, we have defined a different service for each AC task (collector, analyzer, decider, and changer). Also, we have defined a *knowledge* element that represents the knowledge that has the manager of its environment and that is built from the events received, the results of the analysis, the application of policies, etc.

Once we have defined the services we deploy them in top of GEMBus [16], a framework developed inside the GÉANT project to provide a new environment to enable users to create, integrate, and request service facilities on demand by means of the expansion of the current service model to produce the basic framework for a Multi-Domain ESB (MDESB). The inclusion of AC services into GEMBus framework may provide valuable capabilities to applications and services already deployed on GEMBus framework. Moreover, when the autonomic management solution is built in top of GEMBus it can interact with other management solutions, such as AutoBAHN [3] and PerfSONAR [10] as we show in the following section.

Below we show a brief description of each necessary component to assembly the whole solution but before we want to illustrate how it works:

1. The collector receives messages that can be sent from different sources containing one or more registers to describe (part of) the current environment state. These messages are used to build the environment description set (knowledge) and sent to the analyzer. If there is no message received from the outside, a special element built with the Quartz Service Engine sends periodic messages to keep the knowledge alive.
2. The analyzer is built with a Complex Event Processor (CEP) based on Esper [6] so it is capable to detect situations comprising several knowledge

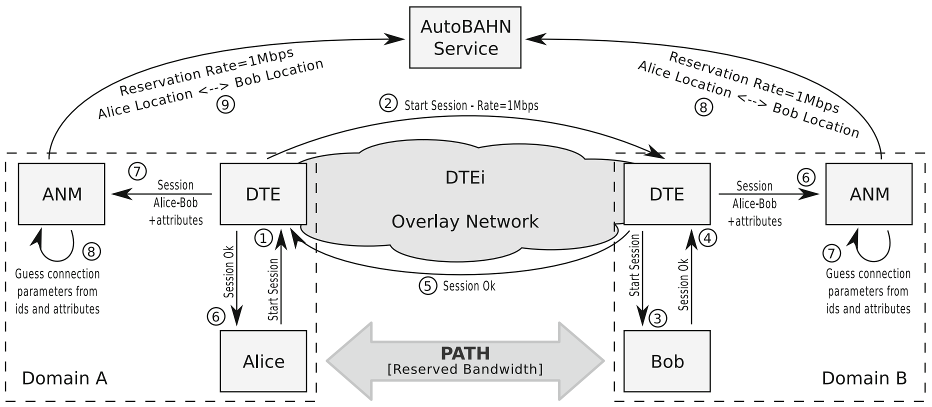
AutoBAHN Service
Reservation Rate=1Mbps Alice Location <--> Bob Location 9
Reservation Rate=1Mbps Alice Location <--> Bob Location 8
2 Start Session - Rate=1Mbps
ANM
7 Session Alice-Bob +attributes
DTE
DTEi Overlay Network
DTE
Session Alice-Bob +attributes 6
ANM
8 Guess connection parameters from ids and attributes
7 Guess connection parameters from ids and attributes
1 Start Session
6 Session Ok
5 Session Ok
Start Session 3
4 Session Ok
Alice
PATH [Reserved Bandwidth]
Bob
Domain A
Domain B

**Fig. 3.** Example scenario of interactions of the proposed solution with AutoBAHN service.

items. Thus, it analyzes the environment information and extracts new or updated items that are then sent back to the collector to complete the knowledge set.

3. For each received message, either from the outside, the *keepalive* service, or from the analyzer, the collector composes a new knowledge message and sends it to the decider.
4. With the knowledge it has received, the decider checks policies to know the environment correctness and with the result composes a new message to be sent to the changer. The policies are checked against a policy manager provided by an external service that implements a policy administration and decision point (PAP, PDP). It is based on XACML and here we decided to incorporate the implementation offered in XACML-Light [9]. This service is configured by administrators using its own interface to manage XACML policies.
5. The changer receives orders from the decider and communicates the actions (obligations) determined by the policies to the elements that should perform them.

This process shows that the key points of the architecture are the analyzer and the decider. They must be configured by administrators to determine the behavior of the manager but then it are designed to run by itself without other human intervention.

The services are deployed in an Enterprise Service Bus (ESB) which is used as service container and communication bus. It hosts the message router that is used to deliver messages among components. Here we use FUSE ESB because it is a standards based, free open-source software and is actively supported. Moreover, the service container offered by the ESB provides a complete component model and life-cycle management tool.

## 4 Evaluation and Results

To show the behavior of the integrated architecture we first define an example scenario that illustrates how the management solution knows the bandwidth requirement of a session and how it contacts with the network management service responsible of performing the bandwidth reservation. Then, we describe the experimental management solution we used to get an approximation of the performance and behavior of the proposed architecture.

First of all, Fig. 3 shows the example scenario. On it, two entities initiate a communication and the ANM sets the communication path, calculating the parameters from the description of the session, which includes the attributes of the identities of both communication parties and the objective of the communication, and the policies that has been set by network administrators.

In the example, steps from 1 to 5 are necessary to establish the communication through the *Identity Plane*. On these steps, Alice requests to start a session with Bob, specifying both identities and the aim of this session. Then, the DTEi instances talk to each other in order to negotiate the session. During this negotiation, Bob is actually asked to start the session and it accepts, so its DTEi instance accepts the negotiation and Alice's DTEi instance communicates it to Alice.

Once the communication has been accepted, the DTEi instances report to their ANMs that such session has been started, including the identities involved with their relevant attributes and the aim of the session. Then, the ANM checks the policies that apply to such identities and communication objective to determine the action to take. In the example, the action told and accepted by the policies implicated in such operation is to contact the AutoBAHN service in order to reserve a bandwidth of 1 Mbps between Alice and Bob. The AutoBAHN service here plays the role of the SDN controller because bandwidth reservation is performed on top of the SDN controller, as a network service or application running on it.

This also demonstrates the great benefit of deploying the management architecture in top of GEMBus, because it wins easy access to many network services, like AutoBAHN. We should notice that since the *Identity Plane* does not know the locators of the entities that start the communication until it has been accepted, the reservation request can not be fired up before receiving the *Session OK*. However, other scenarios may benefit from such action, so DTEi may be configured to trigger more events.

Once the scenario has been defined and the experimental implementation has been built, we have evaluated the solution by the generation of arbitrary events at different rates and the measurement of the time spent from the reception of a message (event) to the emission of a response (decision). We get the measures directly from the host where resides the management solution to avoid any latency that could be introduced by the network. The results are shown by Table 1 and we discuss them below.

In Fig. 4 we compare the average and total times spent to process each event while increasing the number of concurrent events. The average time is taken from the reception of the first message to the emission of the final decision, but

**Table 1.** Performance results (milliseconds).

| Concurrent sessions | 1 | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|---|
| Average time | 18.700 | 24.600 | 40.100 | 78.531 | 147.156 | 312.625 |
| Total time, average | 18.700 | 48.700 | 79.100 | 176.250 | 297.500 | 701.000 |
| Total time, median | 18.000 | 41.000 | 86.000 | 175.500 | 297.500 | 701.000 |
| Normali. total time | 18.700 | 24.350 | 19.775 | 22.031 | 18.594 | 21.906 |
| Ses. estab. overhead | 16.775 | 16.674 | 8.339 | 4.170 | 3.127 | 1.042 |

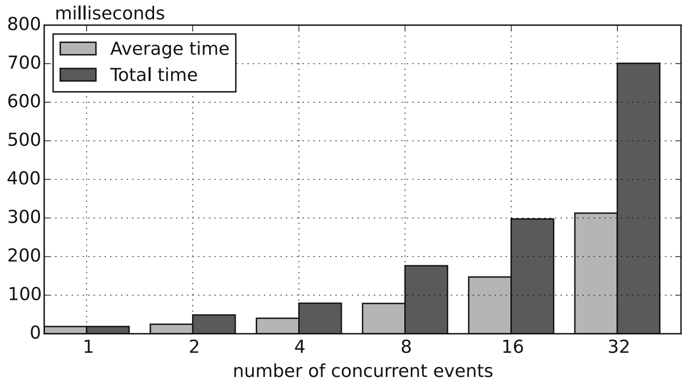

**Fig. 4.** Performance overview showing the average and total times spent processing each event.

the total time is the time spent in processing all concurrent events. Although the average time increases exponentially with the number of concurrent events, this does not mean poor scalability because the events are processed in series, thus there are many events processed at the same time but in different stage of the process. This parallelism is proved watching the total time spent in processing all messages. The total time does not match with the sum of the time spent in each individual event (or the multiplication of the average time spent processing an event by the total number of events being processed). On the contrary, we can see that the total time is around the double of the average time, what demonstrate the high level of parallelism and concurrent behavior of the architecture.

Finally, in Fig. 5 we show a correlation of the manager load, represented in number of threads, and the average time spent in event processing, from the reception of the first message to the emission of the final decision. We obtain this correlated time by dividing the average time spent in event processing by the number of concurrent events being processed at the same time. While concurrency level increases, the correlated time converges to 10 ms. This means that a manager processing 32 events at the same time is going to spend an average of 320 ms to each event, that is the product of 10 ms by 32 concurrent events.

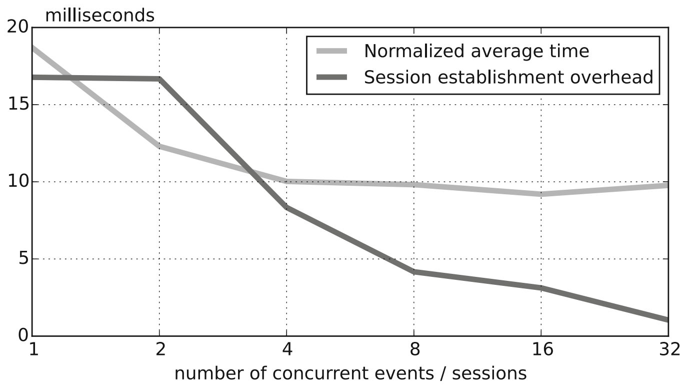

**Fig. 5.** Scalability overview.

Thus, we can infer that if a manager average load reaches 64 concurrent events, each event takes an average of 640 ms to be processed, if it reaches 128 concurrent events, each event takes an average of 1280 ms, and so on. Moreover, the figure shows the evolution of the overhead induced to session establishment by the identity-based architecture per each concurrent session, which states that it spends less than 17 ms to establish single sessions and around 33 ms in all parallel sessions, regardless of the number of concurrent sessions, thus demonstrating the scalability of the architecture proposed here.

## 5 Current Autonomic Management Proposals

In this section we discuss some interesting proposals for autonomic network management that were the starting point to design our solution. An important feature is the service orientation because SOAs are better suited for the specific requirements of our solution. Thus, we briefly comment their strengths and weaknesses, with special attention on the capabilities we require but they lack.

A proper autonomic management solution must provide, at least, transparent support to build distributed systems that involve many elements from different administrative domains, as well as the ability to easily integrate the architecture with existing systems, services, and applications, being or not service oriented and supporting different platforms. Moreover, it should exploit the control-loop concept of AC while offering fine granularity in management tasks. We have found and analyzed some reference solutions that meet most requirements, but with some lacks that justify the architecture presented in this paper.

First we have ANEMA [5]. It is an autonomic network management architecture that is driven by many types of policies and target goals. It follows AC principles and incorporates many network level elements and functions, but lacks the definition of specific mechanisms to involve multiple domains in the management process. Also, its mechanisms are not clearly provided as generic and reusable components, so it is difficult to customize it and make it interact with other external systems.

From the web service point of view, we have PAWS [2], that is a framework to build self-managed applications based on adaptive web services and following both AC and SOA principles. Its main purpose is to enhance business process execution language (BPEL) service composition adding self-configuration and self-healing capabilities. Although this architecture is generic and flexible enough to cover many management requirements, it does not offer a complete AC control loop nor the necessary mechanisms to extend the self-management capabilities out of the service scope.

MAWeS [18] is a new architecture for building service oriented systems that follows SOA principles to provide self-tuning capabilities using automatically generated performance predictions. Although it provides many interesting capabilities, it lacks the specific definition of AC tasks and misses the AC control loop. Also, this architecture does not define how to make services from different domains collaborate to reach distributed objectives.

Finally we have ASMF [4], a framework that follows SOA and AC principles to provide dynamic service composition and enforcement of SLA contracts compliance. Although this architecture is very interesting, it is very tied to service level management and lacks certain interesting features such as the component generalization to permit the customization of the self-management operations, the definition of a policy decision point (PDP) and policy administration point (PAP), and the definition of elements to achieve cross-domain management.

## 6 Conclusions and Future Work

In this paper we have discussed an approach to bring self-management features to the network by using a newly defined *Identity Plane* together with the management model defined by AC, both applied to SDN. Even though current architecture proposals follow AC principles in one level or another, they lack important features to build distributed systems in general, and cross-domain, federated systems in particular. Thus, we proposed to use our simple but complete solution to overcome the challenge but following the same AC principles as the other architectures.

Moreover, we have defined how to perform the integration, the connection points between the management functional block and the SDN controller, and the connection points between the *Identity Plane* and the management solution. In addition, we have demonstrated how the integrated architecture can be used and its good behavior by running an experimental implementation over an example scenario.

Our future work will be focused on the new features that the SDN may offer to the management solution, as well as the evolution of the *Identity Plane* in order to deepen its integration with SDN approaches.

**Acknowledgments.** This work is partially supported by the Ministry of Education of Spain under the FPU program grant AP2010-0576, by the Ministry of Economy and Competitiveness of Spain under the grants TIN2013-50477-EXP and TIN2014-52099-R (Edison project), and by the Program for Research Groups of Excellence of the Séneca Foundation under grant 04552/GERM/06. Finally, this work makes use of results produced by the SmartFIRE project, which is supported by the International Research &Development Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (MSIP, Korea) (Grant number: K2013078191) and the Seventh Framework Programme (FP7) funded by the European Commission (Grant number: 611165).

## References

1. Al-Shaer, E., Greenberg, A., Kalmanek, C., Maltz, D.A., Ng, T.S.E., Xie, G.G.: New frontiers in internet network management. ACM SIGCOMM Comput. Commun. Rev. **39**(5), 37–39 (2009)
2. Ardagna, D., Comuzzi, M., Mussi, E., Pernici, B., Plebani, P.: Paws: a framework for executing adaptive web-service processes. IEEE Softw. **24**(6), 39–46 (2007)
3. Bandwidth on Demand with AutoBAHN. http://www.geant2.net/server/show/ConWebDoc.2544
4. Cheng, Y., Leon-Garcia, A., Foster, I.: Toward an autonomic service management framework: a holistic vision of soa, aon, and autonomic computing. IEEE Commun. Mag. **46**(5), 138–146 (2008)
5. Derbel, H., Agoulmine, N., Salaün, M.: Anema: Autonomic network management architecture to support self-configuration and self-optimization in IP networks. Comput. Netw. **53**(3), 418–430 (2009)
6. Espertech Inc. and Esper contributors. Esper - Complex Event Processing (2010). http://esper.codehaus.org
7. European Future Internet Portal. Future Internet Assembly (2010). http://www.future-internet.eu/home/future-internet-assembly.html
8. Galis, A., et al.: Position Paper on Management and Service-aware Networking Architectures (MANA) for Future Internet (2010). http://www.future-internet.eu/fileadmin/documents/prague_documents/MANA_PositionPaper-Final.pdf
9. Gryb, O., et al.: XACML Light (2010). http://xacmllight.sourceforge.net
10. Internet2, GÉANT2, ESnet, and RNP. perfSONAR: PERFormance Service Oriented Network monitoring ARchitecture (2010). http://www.perfsonar.net
11. Jain, R.: Internet 3.0: ten problems with current internet architecture and solutions for the next generation. In: Proceedings of Military Communications Conference, pp. 1–9. IEEE Computer Society, Los Alamitos, CA, USA (2006)
12. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Comput. **36**(1), 41–50 (2003)
13. Martinez-Julia, P., Gomez-Skarmeta, A.F.: A novel identity-based network architecture for next generation internet. J. Univ. Comput. Sci. **18**(12), 1643–1661 (2012)
14. Martinez-Julia, P., Gomez-Skarmeta, A.F.: Using identities to achieve enhanced privacy in future content delivery networks. Comput. Electr. Eng. **38**(2), 346–355 (2012)
15. Martinez-Julia, P., Gomez-Skarmeta, A.F., Girao, J., Sarma, A.: Protecting digital identities in future networks. In: Proceedings of the Future Network and Mobile Summit 2011, pp. 1–8. International Information Management Corporation (2011)

16. Martinez-Julia, P., Lopez, D.R., Gomez-Skarmeta, A.F.: The gembus framework and its autonomic computing services. In: Proceedings of the International Symposium on Applications and the Internet Workshops, pp. 285–288. IEEE Computer Society, Washington, DC, USA, 2010
17. Pras, A., Schonwalder, J., Burgess, M., Festor, O., Perez, G.M., Stadler, R., Stiller, B.: Key research challenges in network management. IEEE Commun. Mag. **45**(10), 104–110 (2007)
18. Rak, M., Villano, U., Mancini, E.P.: Autonomic composite-service architecture with mawes. In: Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems, pp. 1050–1056. IEEE Computer Society, Washington, DC, USA (2010)
19. Schonwalder, J., Fouquet, M., Rodosek, G., Hochstatter, I.: Future internet = content + services + management. IEEE Commun. Mag. **47**(7), 27–33 (2009)