Evolution and Challenges of Software Defined Networking

Ángel Leonardo Valdivieso Caraguay, Lorena Isabel Barona López,
Luis Javier García Villalba, *Senior Member, IEEE*

***Abstract*—Software Defined Networking (SDN) proposes the separation of the control plane from the data plane in network nodes. Furthermore, Openflow architecture through a centralized control of the packet forwarding engines enables the network administrators to literally program the network behavior. The research and results of experiments show clear advantages over traditional network architectures. However, there are open questions to be solved in order to integrate SDN infrastructure and applications in production networks. This paper presents an analysis of the evolution of SDN in recent years. Additionally, this piece of work also describes some interesting SDN/Openflow research initiatives and applications. Finally, there is a disscussion on the main challenges of this new technology.**

***Index Terms*—Future Internet, Software Defined Networking, OpenFlow.**

I. Introduction

Network technologies have become in a critical element in almost all human activity. The number of devices as well as the amount of traffic moving in Internet is growing exponentially. However, the development of new protocols and services (*mobility, VoIP, QoS, video streaming*) has been limited by the hard union between specialized packet forwarding hardware and operating systems running hundreds of static protocols (some of them private) and only allowing the network administrator only the configuration of the network equipment. Additionally, the deployment time of a new idea from the design, simulation, testing, publication in a standard and finally the installation in network equipment can take some years. Clearly, the new generation networks need to change the rigidity of actual network architectures.

Software Defined Networking (SDN) is an architectural innovation that proposes the separation of *control plane* and the *data plane* enabling their independent evolution and development. Ethane [1] was one of the first SDN initiatives. This model was proposed for enterprise networks and uses a centralized control architecture. However, this implementation required the deployment of custom switches (OpenWrt, NetFPGA, Linux) to support Ethane Protocol.

Today, the principal crystallizing of SDN is Openflow [2], an open architecture initially developed as a way of allowing researchers to run experiments on heterogeneous networks without affecting the real users traffic. Openflow [2] aims at providing more configuration options in the switch dataplane.

The Openflow specification [3] establishes the rules of communication between *data plane* and a centralized *control plane* and allow the entire network to be controlled through users software applications (APIs). The Open Networking Foundation ONF [4] brings together about 90 companies and is dedicated to releasing, promoting and adopting of OpenFlow specification [3].

This implementation of Openflow [3] removes the limitation of rigidity of static protocols, opens the possibility of rapid innovation and led research community to develop new paradigms in network technologies. Some examples of this evolution are: Virtualization [5], [6], High Level Network Programming Languages [7], [8], optimization of Quality of Experience QoE applications [9], [10]. However, new challenges and unanswered questions appear when trying to implement these ideas in production networks: Modeling and Performance [11], Resilience and Recovery [12], Security [13], Convergence and Management [14], [15]. This paper presents an analysis of SDN/Openflow [2] technologies and discusses the challenges and open question in order to implement them in production networks.

The rest of the paper is outlined as follows: Section II defines the Openflow architecture. Then, Section III presents the actual research and experimentation. Next, Section IV analyzes the SDN challenges. Finally, Section V concludes with the discussion and conclusions.

II. OPENFLOW

The Openflow architecture [2] consists of three principal elements: an Openflow switch, an external controller and the Openflow Protocol [3] which establishes the communication between switch and controller through a secure channel.

An Openflow switch consist of one or more *flow tables* and a *group table* used for packet lookup and forwarding. Each *flow table* is composed of a set of match fields, counters and instructions. The packet fields to be compared with the match fields can be indistinct of the second, third or fourth layer in TCP /IP architecture. The number of the packet fields to be matched depends on the version of the Openflow protocol [4], [16]. The recent version of Openflow is v1.3 and specifies a number of 40 match fields including support to IPv6. However, the most widely used version is v1.0 with a number of 12 match fields.

The authors are with the Group of Analysis, Security and Systems (GASS), http://gass.ucm.es/en, Department of Software Engineering and Artificial Intelligence (DISIA), School of Computer Science, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain (e-mail: {angevald, lorebaro, javierv}@ucm.es).

If the header of an incoming packet is similar to a match field in a *flowtable*, the corresponding actions are taken. The Openflow Protocol [3], [4] specifies the set of mandatory and optional actions that a switch can take. The principal mandatory actions are: send the packet by a given port, send packet to the controller or drop the packet. In the case that incoming packet doesnt match with a match field, the switch can be configured to send it to the external controller or drop the packet. The external controller has the responsibility of analyzing the information received by switches and controlling their actions through the maintenance of the *flow tables* in switches.

III. RESEARCH AND EXPERIMENTATION

The direct manipulation of the forwarding plane on network devices allows researchers the possibility of designing new services, routing protocols, security models, and directly testing their functionality directly in the network. Next, there is a revision of important SDN research initiatives.

A. Virtualization

Similar to computer virtualization layer, where a hardware abstraction permits slicing and sharing the hardware resources with different Operating Systems OS in a host, the goal of network virtualization is to isolate multiple logical networks, each of them with completely different addressing and forwarding mechanism, but sharing the same physical infrastructure. SDN provides the opportunity to reach a network hardware abstraction layer.

Flowvisor [5] is a virtualization platform that uses Openflow [2] to sit logically between control and forwarding paths. Flowvisor [5] acts as a transparent proxy between controllers and switches. Then, it creates strong and transparent *virtual slices* according to administrators established policies and ensures isolation in terms of bandwidth, flowspace and switch CPU load. The user can only observe and control only their own slice. Additionally, it is possible to slice a *virtual slice* and create like hierarchies of virtualized networks (Fig.1).

In [6] there is a demonstration of four successful networking experiments (load balancer, wireless streaming video, traffic engineering and hardware prototyping experiment) using Flowvisor [5], each with its own slice. However, the experiments show some problems to be solved: the unexpected interaction with other deployed network devices, increase of broadcast traffic emitted from non-OpenFlow devices and some violations of the CPU isolation, especially when one slice inserts a forwarding rule that is handled via the switches slow path.

Another important aspect is the integration between network operations and computer virtualization. In computer virtualization, the different virtual machines VMs require a network access layer that provides inter- and intra-VM connectivity and provide similar network functions to the physical layer. The common model is to establish communication between virtual nodes and physical NIC implementing typical L2 switching or L3 routing. This makes the network management of virtual environments difficult, for example the migration of VMs between different physical servers. In this approach, SDN and network virtualization can help to achieve these goals.

Open vSwitch [17] is a software switch built for virtual environments. This switch exports an interface for fine-grained network control. Additionally, it has a logical partition of the forwarding plane through a flexible, table-based forwarding engine. The forwarding plane has an external interface and can be managed for example by Openflow [2]. With this abstraction, the controller can obtain a logical view of multiple Open vSwitches running on separate physical servers.

Another interesting application of virtualization is the *virtual network migration (VNM)*. In typical networks, the migration or change of a network node require the re-configuration and the re-synchronization of the routing protocol. This causes high delays and packet loss. Consequently, the use of virtual nodes can significantly reduce the downtime.

In the VNM system proposed in [18], the SDN controller creates new flow entries for the new switch and redirects the paths from the initial to the new node. Then, the controller deletes the flow entries of the old switch making it feasible to be removed with security. The results of experiments show a total migration time of 5ms without packet loss. Moreover, the system could be dynamically reconfigured to place virtual networks on different physical nodes according to the time of day or the traffic demand to save energy (green networks).

B. Simulation and Testbeds

The possibility of debugging and testing new designs in a real environment is difficult and expensive. The network simulation is the first step to evaluating the idea previous implementation in a real production network. However, the current network simulators are expensive and dont offer support to Openflow [2].

Mininet [19] is the first open source simulator for Openflow networks. It uses OS-level virtualization features, including processes and network namespaces and allows a scale to hundreds of nodes (switches, hosts, and controllers) in a simple laptop. The user can run commands on hosts, verify switch operation and even induce failures or adjust link connectivity. Another advantage is that the code used to program a controller in the simulator is the same as the one to be deployed into a real network. However, Mininet [19] presents some limitations. The performance is directly related to the number and topology of virtual nodes and the available resources in the host. For this reason, the results of experiments in terms of bandwidth or time can vary from one computer to another.

In case the industry or research community may need to test experiments in a scaled real network infrastructure, there is worldwide in development different large scale testbeds. Global Environment for Network Innovations GENI [20] is a research infrastructure sponsored by the US National Science Foundation. Currently GENI [20] is deploying a SDN/Openflow [2] technology at their infrastructure (8 campuses interconnected).

The project OpenFlow in Europe: Linking Infrastructure and Applications OFELIA [21] bring together different European research institutes within the European Commisions FP7 ICT Work Programme. This project interconnects an infrastructure of 8 Openflow [2] islands allowing experimentation on multi-layer and multi-technology networks and providing realistic test scenarios and diverse and scalable resources. The Extending and Deploying Ofelia in BRAzil EDOBRA proposal [22] has the purpose of extending the OFELIA network to Brazil as a new OFELIA island.

C. High Level Network Policy Languages

In computer systems, the Operating-Systems (OS) facilitate the development of programs through high level abstractions of hardware and information resources. In the SDN paradigm, the entire behavior of the network is managed by the control layer. In the Openflow architecture [2], the control layer is materialized in a centralized controller. Clearly, the controller needs a Network Operating System NOS that facilitates the user create software programs to control the behavior of the network. Currently, there are several NOS. The most used open source NOS are: NOX/POX [23], Maestro [24], Beacon [25], Floodlight [26] each with its particular characteristics. NOX and POX are similar but implemented in C++ and Python respectively. Additionally, they use a group of application programming interfaces (APIs) to communicate and interact with switches. Beacon [25] and Floodlight [26] are NOS based on Java platform. Maestro [24] use multithreading looking for better performance and scalability. In [27] these kinds of controllers are denominated as *southbound*, because they interact directly with the forwarding layer and manipulate the state of individual devices. However, programming High Level operational tasks using this kind of NOS is still difficult. The user need to mix different match fields, actions, prorities for each switch looking for the espected behavior of the network. Moreover, the synchronization between control and data messages requires high attention and complicates the creation of new applications. For this reason, the *northbound* interfaces automatically translates the network policies (policy layer) in coordinated rules to the switches and ensure a correct behaviour of the network.

Procera [8] is a control architecture that includes a declarative policy language based on the notion of *Functional Reactive Programing (FRP)* [28] which provides a declarative, expressive and compositional framework for describing reactive and temporal behaviors according to network conditions. It also has event comprehensions to manipulate event streams and signal functions of windowing and aggregation. In [27], [29] uCap project is presented using Procera [8]. uCap [29] enables home users to monitor and manage end-host devices data usage in their home network. When the device reaches the data capacity allocated by the administrator, the system can disable the connection of this device. This system is useful especially for subscribers of ISPs that enforce monthly bandwidth caps.

Frenetic [7] is a network programming language that comprises two integrated sublanguages: a *Declarative Network Query Language* and a *Network Policy Management Library*. The *Network Query Language* reads the state of the network by installing low-level rules on switches that are transparent to the programmer. The *Network Policy Management Library* manages the policy of the forwarding packets using a functional, reactive programming based on FRP [28]. A working prototype of Frenetic is available in [30].

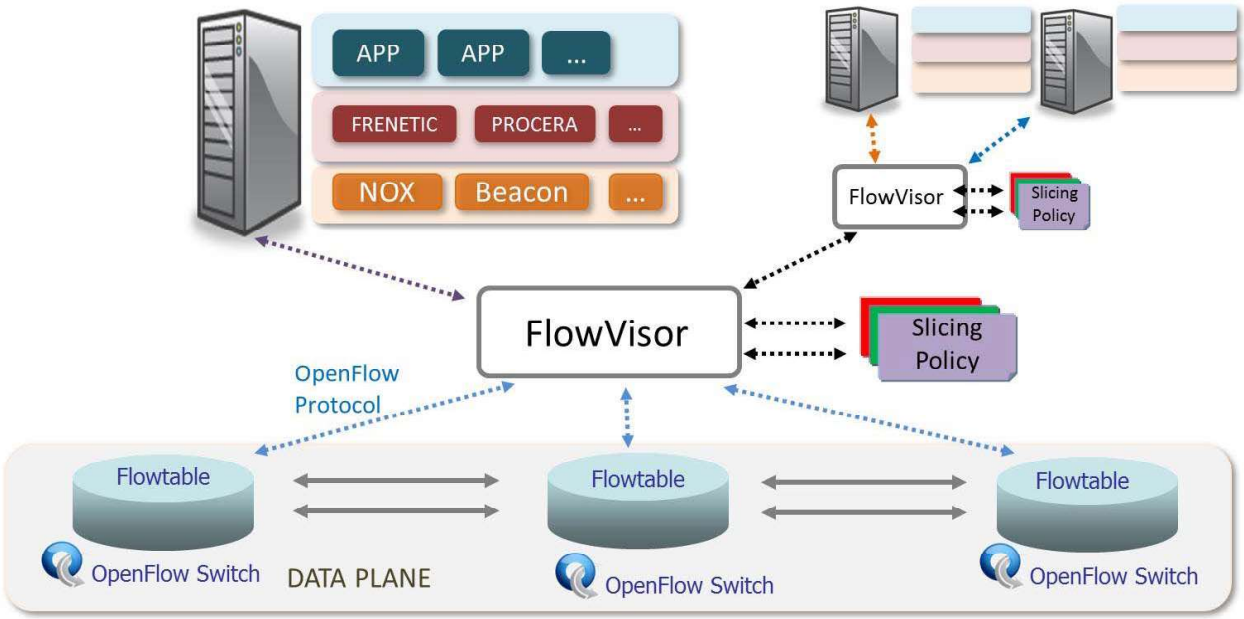

Fig. 1. Openflow Protocol, Virtualization and Network Operating Systems NOS

Another High Level Policy Language is Pyretic [31], which is designed to support modular programming. In other words, the programmer can write modules independently without interferance between them. Pyretic [31] establish different policy composition operators, including *parallel composition* and *sequential composition*. In *parallel composition* the policies are executed on the same packet and the results are combined. For example, send packet with header A to port 1 and packet with header B to port 2. In *sequential composition* the result of a policy A is the entry of the next policy B, such as the combination of balancing and switching policies (change the packet header and then routing based on the new header). The Fig.1 describes the location and function of the different abstraction levels (Openflow [2], Virtualization and NOS) within SDN paradigm.

D. Multimedia - Quality of Experience (QoE)

In multimedia systems, the notion of Quality of Experience QoE has growing since Quality of Service (QoS) is not powerful enough to express all features involved in a communication service. However, the term QoE doesnt yet have a solid, theoretical and practical concept.

The European Network on Quality of Experience in Multimedia Systems and Services (QUALINET [9]) emphasizes a working definition: *QoE is the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the users personality and current state*. Regardless, the use of this term QoE is increasing in the research community.

In [10], a SDN path optimization system for multimedia flows at the infrastructure layer improves QoE experience. The architecture consists of two main functions: QoS Matching and Optimization Function (QMOF) and the Path Assignment Function(PAF). QMOF reads the user multimedia parameters, determines whats feasible and looks for an optimal service configuration, while PAF maintains a network topology database. If the quality of the network is degraded, the system can react and dynamically modify the path parameters of the network prioritizing the users configuration.

Another important goal in multimedia communications and QoE is the capability to recovery a network path when a link fails (eg. *audio/video streaming*). In [32], a Openflow [2] based system is capable of recovering from a link failure using an alternative path. The system uses two well-known mechanisms: *restoration* and *protection*. In *restoration* method, when a path failure signal is received, the controller calculates and establishes an alternative path. In *protection* method, the controller anticipates a failure and calculates two disjoint paths (working and protected) before a failure occurs. The experiments show a recovery time of less than 50 ms for carrier-grade network.

IV. CHALLENGES

The deployment and experimentation of SDN/Openflow [2] initiatives have some important challenges at the moment to be successfully adopted at production networks. Below, there is a discussion of the most important aspects to considerate and the first proposed solutions.

A. Modeling and Performance

The implementation of SDN/Openflow requires the estimation of the number of controllers needed by a determinate topology and the localization of these controllers. This is a difficult question to answer. SDN establishes the separation of control and data planes, but doesnt necessarily order the existence of a single controller. SDN control plane may have single controller, a set of controllers in a hierarchy topology, or even any dynamic controllers topology. In [33], there is a complete study about this issue. The authors conclude that number and position of controllers depends on desired reaction bounds, metric choice and the network topology itself. The experiments show a decreased performance from each added controller, along with tradeoffs between metrics. However, the latency of a node connected with a single controller for medium sized network is similar to the response-time in actual production networks.

Another important factor is the selection of an appropriate NOS and the measure of its performance. In other words, how fast the controller responds to datapath request and how many datapath requests the controller can handle per second. However, it is also necessary to establish a balance between high performance and the productivity of developers (a developer friendly language). The experimentation [25], [34] show that the performance of the controller depends directly of programming language (C++, Java, Python) and the development environment.

The NOS previously named [23]–[26] are still in development and their features are constantly improved. For example, Fig. 2 and 3 show an evaluation of recently new Beacon improvements of performance in relation to other controllers. Tests were run on Amazons Elastic Computer Cloud using a Cluster Compute Eight Extra Large instance, containing 16 physical cores from 2 x Intel Xeon E5-2670 processors, 60.5GB of RAM, using a 64-bit Ubuntu 11.10 VM image (ami-4583572c) [25]. The experiment uses Cbench [35] utility for testing the throughput (number of transactions per second) and latency (time required for a packet to arrive at its destination through the network) of Openflow [2] controllers using a single thread.

Fig.2 shows the capacity of the different NOS handling constantly Packet In messages from 64 emulated switches as soon as possible. Fig.3 measures the average time between sending a single packet of an emulated switch and the reply from the controller.

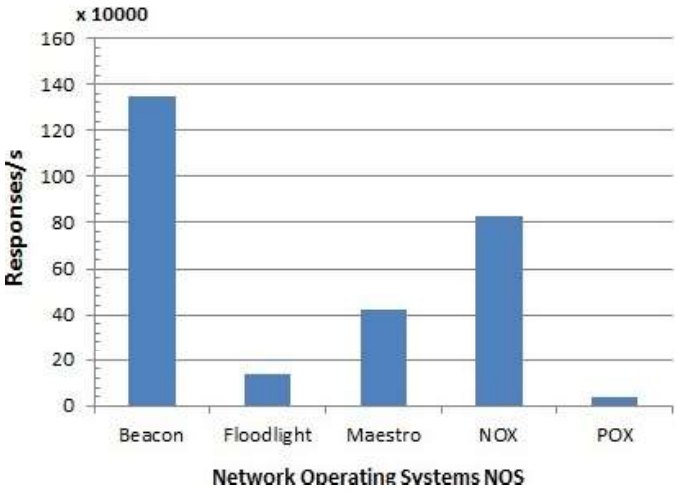

Fig. 2. Test of throughput of NOS [25]

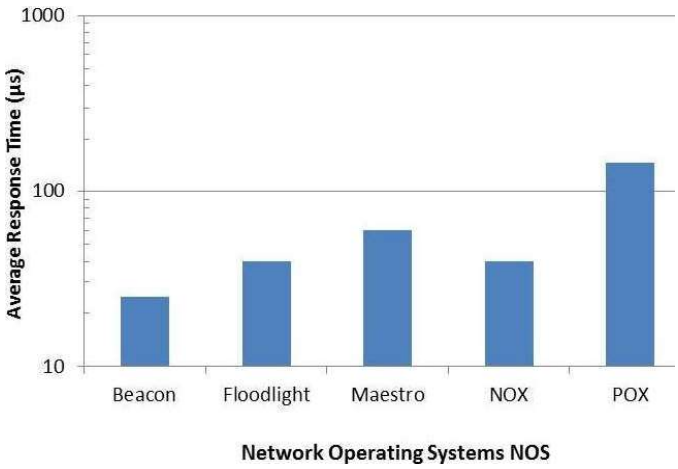

Fig. 3. Test of latency of NOS [25]

B. Resilience and Recovery

One of the problems related with the centralized control proposed by Openflow [2] is the vulnerability of the network. A failure of the controller can negatively compromise resilience of the whole network. Additionally, Openflow [2] establishes the configuration of one or more backup controllers, but doesnt provide a coordination mechanism between them.

CPRecovery component [12] is primary-backup mechanism that permits the replication between primary and secondary SDN controller in two phases: the *replication phase* and the *recovery phase*. The *replication phase* acts during normal functioning of the controller and send regular updates to the backup controller. The *recovery phase* acts during a failure state of the primary controller and starts the backup controller as a main controller. The failure state can be fired by abruptly aborting of a NOS process, an unexpected error of the application (API) or a DDoS attack. The results of experiments show a recovery time (from failure state to backup connection) of 800 ms.

C. Security

The current Internet architecture has huge problems to support, verify and enforce security properties. Typical Security Systems are based on securing hosts (e.g. antivirus) or installing specialized network devices (middleboxes) to detect anomalies into the network. However, these solutions become ineffective when the host is compromised (zero day vulnerabilities) or require constant updates and effort from the network administrator for example selecting the traffic to be filtered. In this point SDN can help to improve and develop new security models.

Pedigree [13] is a security system based on SDN for enterprise networks. This model uses Openflow [2] and a simple kernel OS-function on hosts to help the network controller to determine the provenance of the traffic network. Once the connection is validated, the controller enables and establishes the network path in switches. This system is designed with two components: the *tagger* and the *arbiter*. The *tagger* monitors all events in the OS and creates a tag for each process. When a process requires data sent through the network, a tagstream is sent to the controller. The controller has an *arbiter* function that checks the tags, verifies and orders an installment of the appropriate flowtables in switches. If the arbiter detects an unauthorized connection attempt, the controller orders a blockage of the flow of this link. Although, these processes generate additional load on the host and the network, the overhead is comparable to running an antivirus software.

D. Convergence and Management

SDN-Opeflow [2] architecture was originally developed for enterprise-campuses networks. However, to try and extend this architecture to large networks requires attend some issues, for example the problem of Interdomain Routing.

Today, the largest deployed protocol to communicate autonomous systems AS is the Border Gateway Protocol (BGP). BGP is a destination-based forwarding paradigm. In other words, the routing information between Autonomous Systems AS depends on the destination address in the IP packet header. An AS interchanges information and influence the traffic flow only with his direct neighboring AS. This makes it difficult to control of traffic flows along an entire path or in a remote part of the network. The task is to integrate SDN architecture to interconnect different AS.

In [15], an interdomain-routing solution based on BGP uses a NOX-Openflow [2] architecture. The new Inter-AS system proposes to extend two NOX components: *messenger* and the *switch*. *Messenger* creates a dedicated channel (port 2603) and enables the exchange of information between several NOX components. In other words, this *messenger channel* is used as a dependency of the Inter-AS component. The *Switch* adds a call to the Inter-AS component in case of an outside destination and selects the right datapath and port to forward the packet.

Another challenge to considerate is the integration with legacy networks. That is, networks with non-Openflow capacity (actual Internet infrastructure). It is noted that equipment could be upgraded or even replaced to support Openflow [2]. However, this change means costly network re-engineering. For this reason, the coordination between new SDN equipment and legacy infrastructure is necessary.

The project LegacyFlow [36] proposes a solution for coordination between Openflow [2] and traditional networks. The model introduces a new component between these architectures called *Legacy Datapath(LD)* that provides a bridge between different features of the legacy equipment. LD receives and interprets the messages sent by the Openflow controller and applies the corresponding actions in a real switch. Additionally, LegacyFlow [36] also proposes to add new actions to Openflow protocol [3] in order to improve the integration with traditional networks.

V. DISCUSSION AND CONCLUSIONS

This paper presents the state-of-the-art, the evolution and the challenges of Sofware Defined Networking in the last few years. Below, there is an exposition of some final comments and conclusions.

The separation of *control* and *data planes* opens the possibility to easily develop new services and network applications to industry and research community. Openflow [2] is a SDN technology based on the logic of match/action in *data plane* taking advantage of the actual network hardware capabilities. However, SDN can be expanded beyond match/action paradigm. For example, SDN can take into account several extensions, such as middleboxes or programmable custom packet processors.

This evolution could offer new services, for example on the fly encryption, transcoding, or traffic classification. Nevertheless, this requires a unifying control framework to allow coordination between the different types of network devices, as well as consensus and vendor support. At this point, the ONF [4] will take a decisive factor with the publication of new SDN protocols.

In *control plane*, the programming, debugging and testing are open issues in SDN. The High Level Languages facilitate the development of applications, but the composing and coupling of heterogeneous component are still difficult. For example, compose applications using NOX/POX [23] and Floodlight [26] or Beacon [25] simultaneously in the same controller is complicated today. The northbound policy languages also need to be constantly improved and tested to their implementation in production networks.

Finally, it is necessary to remark that Software Defined Networking is a tool. The research community can use this tool and develop new protocols, services, network applications taking advantage of the global view and the huge amount of information about the network.

ACKNOWLEDGMENT

This work was supported by the Agencia Española de Cooperación Internacional para el Desarrollo (AECID, Spain) through Acción Integrada MAEC-AECID MEDITERRÁNEO A1/037528/11.

Ángel Leonardo Valdivieso Caraguay and Lorena Isabel Barona López are supported by the Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación SENESCYT (Quito, Ecuador) under Convocatoria Abierta 2012 Scholarship Program N. 2543-2012. The authors would also like to thank Ana Lucila Sandoval Orozco for her valuable comments and suggestions to improve the quality of the paper.

REFERENCES

[1] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking Control of the Enterprise," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, (New York, NY, USA), pp. 1–12, ACM, August 2007.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 69–74, March 2008.

[3] "OpenFlow Switch Specification v.1.1.0," pp. 1–56, 2011.

[4] "Open Networking Foundation." https://www.opennetworking.org/.

[5] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A Network Virtualization Layer," tech. rep., OpenFlow Switch Consortium, October 2009.

[6] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the Production Network be the Testbed?," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, (Berkeley, CA, USA), pp. 1–6, USENIX Association, October 2010.

[7] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A Network Programming Language," in *Proceedings of the The 16th ACM SIGPLAN International Conference on Functional Programming*, (New York, NY, USA), pp. 279–291, ACM, September 2011.

[8] A. Voellmy, H. Kim, and N. Feamster, "Procera: A Language for High-Level Reactive Network Control," in *Proceedings of the First Workshop on Hot topics in Software Defined Networks*, (New York, NY, USA), pp. 43–48, ACM, August 2012.

[9] S. M. Patrick Le Callet and A. Perkis, "Qualinet White Paper on Definitions of Quality of Experience," *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, March 2012.

[10] A. Kassler, L. Skorin-Kapov, O. Dobrijevic, M. Matijasevic, and P. Dely, "Towards QoE-driven Multimedia Service Negotiation and Path Optimization with Software Defined Networking," in *Proceedings of the 20th International Conference on Software, Telecommunications and Computer Networks*, vol. 1, pp. 1–5, IEEE, September 2012.

[11] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and Performance Evaluation of an OpenFlow Architecture," in *Proceedings of the 23rd International Teletraffic Congress*, pp. 1–7, ITCP, September 2011.

[12] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A Replication Component for Resilient OpenFlow-based Networking," in *Proceedings of the 2012 IEEE Network Operations and Management Symposium*, pp. 933–939, IEEE, April 2012.

[13] A. Ramachandran, Y. Mundada, M. B. Tariq, and N. Feamster, "Securing Enterprise Networks Using Traffic Tainting," tech. rep., August 2009.

[14] S. Das, G. Parulkar, N. McKeown, P. Singh, D. Getachew, and L. Ong, "Packet and Circuit Network Convergence with OpenFlow," in *2010 Conference on Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference (OFC/NFOEC)*, pp. 1–3, IEEE, March 2010.

[15] R. Bennesby, P. Fonseca, E. Mota, and A. Passito, "An Inter-AS Routing Component for Software-Defined Networks," in *Proceedings of the 2012 IEEE Network Operations and Management Symposium*, pp. 138–145, IEEE, April 2012.

[16] F. de Oliveira Silva, J. de Souza Pereira, P. Rosa, and S. Kofuji, "Enabling Future Internet Architecture Research and Experimentation by Using Software Defined Networking," *European Workshop on Software Defined Networking*, vol. 0, pp. 73–78, October 2012.
[17] B. Pfaff, J. Pettit, K. Amidon, and M. Casado, "Extending Networking into the Virtualization Layer," in *Proceedings of ACM SIGCOMM HotNets*, ACM, October 2009.
[18] P. S. Pisa, N. C. Fernandes, H. E. Carvalho, M. D. Moreira, M. E. M. Campista, L. H. M. Costa, and O. C. M. Duarte, "OpenFlow and Xen-Based Virtual Network Migration," in *Communications: Wireless in Developing Countries and Networks of the Future*, vol. 327, pp. 170–181, Springer Berlin Heidelberg, September 2010.
[19] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for. Software-Defined Networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, (New York, NY, USA), pp. 19:1–19:6, ACM, October 2010.
[20] C. Elliott, "GENI: Opening Up New Classes of Experiments in Global Networking," *IEEE internet computing*, vol. 14, pp. 5–10, January 2010.
[21] "OFELIA." http://www.fp7-ofelia.eu/.
[22] C. Guimarães, "Extending and Deploying Ofelia in BRAzil (EDOBRA)," February 2013.
[23] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an Operating System for Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 105–110, July 2008.
[24] T. S. E. N. Zheng Cai, Alan L. Cox, "Maestro: A system for scalable openflow control," tech. rep., December 2010.
[25] D. Erickson, "The Beacon Openflow Controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot topics in Software Defined Networking*, (New York, NY, USA), pp. 13–18, ACM, August 2013.
[26] "Floodlight." http://www.projectfloodlight.org/.
[27] H. Kim and N. Feamster, "Improving Network Management with Software Defined Networking," *IEEE Communications Magazine*, vol. 51, pp. 114–119, February 2013.
[28] Z. Wan and P. Hudak, "Functional Reactive Programming from First Principles," in *Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation*, (New York, NY, USA), pp. 242–252, ACM, May 2000.
[29] H. Kim, S. Sundaresan, M. Chetty, N. Feamster, and W. K. Edwards, "Communicating with caps: Managing Usage Caps in Home Networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, (New York, NY, USA), pp. 470–471, ACM, August 2011.
[30] "Frenetic." https://github.com/frenetic-lang/frenetic.
[31] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing Software Defined Networks," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, (Berkeley, CA, USA), pp. 1–14, USENIX Association, April 2013.
[32] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "A Demonstration of Fast Failure Recovery in Software Defined Networking," in *Testbeds and Research Infrastructure. Development of Networks and Communities*, vol. 44, pp. 411–414, Springer Berlin Heidelberg, June 2012.
[33] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, (New York, NY, USA), pp. 7–12, ACM, August 2012.
[34] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On Controller Performance in Software-defined Networks," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, (Berkeley, CA, USA), pp. 10–10, USENIX Association, April 2012.
[35] "Cbench." http://archive.openflow.org/wk/index.php/Oflops.
[36] F. Farias, J. Salvatti, E. Cerqueira, and A. Abelem, "A Proposal Management of the Legacy Network Environment Using Openflow Control Plane," in *Proceedings of the 2012 IEEE Network Operations and Management Symposium*, pp. 1143–1150, IEEE, April 2012.

Ángel Leonardo Valdivieso Caraguay was born in Loja, Ecuador, in 1985. He received a B.S. degree in Electronics and Telecommunications Engineering from the Escuela Politécnica Nacional, Quito, Ecuador in 2009 and a M.S. degree in Information Technology from the University of Applied Sciences Hochschule Mannheim, Germany in 2012. He is currently a Ph.D. student of Computer Engineering in Universidad Complutense de Madrid, Spain. His research interests include computer networks and software-defined networking.

Lorena Isabel Barona López was born in Ambato, Ecuador, in 1985. She received a B.S. degree in Electronic and Information Networks Engineering from the Escuela Politécnica Nacional, Quito, Ecuador in 2010 and a M.S. degree in Telecommunications Engineering from the Universidad Politécnica de Madrid, Spain in 2013. She is currently a Ph.D. student of Computer Engineering in Universidad Complutense de Madrid, Spain. Her research interests include computer communication and software-defined networking.

Luis Javier García Villalba (SM04) was born in Madrid, Spain, in 1969. He received the Telecommunication Engineering degree from the Universidad de Málaga, Spain, in 1993, the M.Sc. degree in computer networks in 1996, and the Ph.D. degree in computer science in 1999, the last two from the Universidad Politécnica de Madrid, Spain. He was a Visiting Scholar at the Research Group Computer Security and Industrial Cryptography (COSIC), Department of Electrical Engineering, Faculty of Engineering, Katholieke Universiteit Leuven, Belgium, in 2000, and a Visiting Scientist at the IBM Research Division (IBM Almaden Research Center, San Jose, CA) in 2001 and in 2002. He is currently an Associate Professor in the Department of Software Engineering and Artificial Intelligence at the Universidad Complutense de Madrid (UCM) and Head of the Complutense Research Group GASS (Group of Analysis, Security and Systems), which is located at the School of Computer Science at the UCM Campus. His professional experience includes research projects with Hitachi, IBM, Nokia, Safelayer Secure Communications, and VISA. His main research interests are in information security, computer networks, and software-defined networking. Dr. Garca Villalba is an Associate Editor in Computing for IEEE Latin America Transactions since 2004.