OMNI: OpenFlow MaNagement Infrastructure

Diogo M. F. Mattos, Natalia C. Fernandes, Victor T. da Costa, Leonardo P. Cardoso,
Miguel Elias M. Campista, Luís Henrique M. K. Costa, Otto Carlos M. B. Duarte
Universidade Federal do Rio de Janeiro - Rio de Janeiro, Brazil
Emails: {menezes, natalia, torres, cardoso, miguel, luish, otto}@gta.ufrj.br

***Abstract*—Managing computer networks is challenging because of the numerous monitoring variables and the difficulty to autonomously configure network parameters. This paper presents the OpenFlow MaNagement Infrastructure (OMNI), which helps the administrator to control and manage OpenFlow networks by providing remote management based on a web interface. OMNI provides flow monitoring and dynamic flow configuration through a service-oriented architecture. OMNI also offers an Application Programming Interface (API) for collecting data and configuring the OpenFlow network. We propose a multi-agent system based on OMNI API that reduces packet loss rates. We evaluate both the OMNI management applications and the multi-agent system performance using a testbed. Our results show that the multi-agent system detects and reacts to a packet-loss condition in less than three monitoring intervals.**

I. Introduction

The OpenFlow platform enables creating test networks in parallel with the production network, using commercial equipment [1]. Hence, since it is an open standard, OpenFlow supports innovation and enables the development of new control mechanisms and experiments in real environments [2]. Nevertheless, OpenFlow demands tools to simplify management and to facilitate the development of control mechanisms.

The OpenFlow architecture defines a network model in which the forwarding elements, called OpenFlow switches, are simple and programmable. A special node, the controller, centralizes the execution of all control tasks. The controller is also used for deploying new mechanisms, such as new routing protocols or optimized cross layer packet-forwarding algorithms. There are controller implementations available for the OpenFlow platform [3]–[5], among which the NOX controller [6], used in our work, stands out.

To decentralize network control and employ multiple controllers, the OpenFlow platform provides FlowVisor [7]. FlowVisor is an application that virtualizes the network control mapping controller actions to network switches and forwarding messages from switches to each correspondent controller. Hence, FlowVisor slices the network into virtual networks and instantiates a single controller for each virtual network.

Managing an OpenFlow network is challenging. OpenFlow, however, provides more flexibility than an Ethernet switched network, because OpenFlow forwards packets based on information of the link layer, the network layer, or even the transport layer, while Ethernet is restricted to forward packets based on link layer information. Thus, there are proposals for explicitly managing OpenFlow networks. OpenRoads is an experimental facility based on OpenFlow [8]. OpenRoads introduces a tool for managing the IP addresses that are assigned to each network slice. OpenRoads also provides graphical information to the network administrator; examples are the average flow duration or the number of active flows in a certain interval. Another proposal for simplifying OpenFlow network management is SNAC (Simple Network Access Control) [4]. SNAC is an OpenFlow controller based on a graphical user interface (GUI) in which the administrator deploys polices and monitors network information. SNAC also implements middlebox functions, a network address translation (NAT) or a captive portal. Thus, these proposals are focused on monitoring, or on deploying a network function on an OpenFlow switch. Our proposal, besides monitoring the network, provides an efficient migration primitive and an autonomic control mechanism.

Management also concerns acting over the OpenFlow network. Wang et al. propose an algorithm for using an OpenFlow network for load balancing [9]. Wang et al. rearrange the flows in order to divide requests from clients into several server replicas, according to the administrator setup. This proposal autonomously reacts to a change in the number of active server replicas, but does not react to other network changes. On another proposal, Kim et al. implement Quality of Service (QoS) primitives on an OpenFlow switch [10]. This proposal allocates resources for QoS-sensible flows to guarantee a minimum bandwidth and a maximum delay bounds. Both proposals act on the network in an automatic fashion, reacting to changes in network conditions, but do not act autonomously by each switch, since a centralized controller that has a global view of the network takes all actions. Differently, in our proposal each switch reacts autonomously, i.e. the switches are able to identify a network fault, reason, and react.

This paper proposes the OpenFlow MaNagement Infrastructure (OMNI)[1], for controlling and managing OpenFlow networks, and also enabling the development of autonomous applications. OMNI provides a set of tools to control and manage the network, a web interface for the administrator to interact with these tools, as well as a multi-agent system to autonomously control the network. Among these tools, two passive tools stand out, one that collects statistics about the flows, and another that probes the network for obtaining the physical network topology. As tools that acts on the network, we provide an interface for creation, modification and deletion of flow forwarding rules, as well as an interface to migrate flows, which allows the

This work was supported by FINEP, FUNTTEL, CNPq, CAPES, Cofecub, and FAPERJ.

[1] http://www.gta.ufrj.br/omni

administrator to change the physical path of a flow on the network. It is worth noting that the migration facility ensures flow migration without packet losses. Thus, the administrator can execute preventive maintenance tasks of switches or use green computing techniques to reduce the number of active switches. OMNI also provides autonomous management facilities by using the Ginkgo agent platform [11].

Therefore, the administrator can program agent behaviors to monitor, detect anomalies and act on the network. In this paper we also evaluate the ability of autonomous agents to detect packet loss events and to react in a short time. Thus, we develop a multi-agent system that detects a traffic congestion condition on a virtual link causing high packet loss rates and ensures that the network does not use this virtual link. The migration evaluation shows that this feature provides low response time, and is performed without breaking connections or losing packets. Our experiments show that the developed agents present a low response time for detecting packet loss issues, and for taking actions to solve the problem.

This paper is organized as follows. In Section II, we describe the OMNI applications that we developed based on NOX controller. Section III describes the OMNI web interface. In Section IV, we explain the developed agent behavior. Section V presents our experimental scenario and results. Finally, Section VI concludes the paper.

II. NOX Applications for OpenFlow Management

The main goals of the OpenFlow MaNagement Infrastructure (OMNI) are to control and to manage an OpenFlow network. The management and control facilities are performed by applications running on top of the NOX controller, which is an application that implements the OpenFlow protocol and acts as an interface between control applications and the network.

One of the OMNI applications is a web service provider, which exports an Application Programming Interface (API) for accessing other applications as web services. We also develop a server to provide a user-friendly web interface for the network administrator. This server communicates with NOX through the exported web service API. Moreover, the multi-agent system also accesses web services provided by NOX, in order to autonomously control the network. Fig. 1 presents the set of NOX applications that form the OMNI tool. We implemented from scratch the `Flow Migration` and the `Flow Manager` applications and we extended existing NOX applications to offer the `SwitchStats`, which is referenced in Fig. 1 as `Stats`, and the `Web Server` applications. We also adapt `Discovery`, `Spanning Tree`, and `PySwitch`, which is a basic application and is not shown in Fig. 1, applications to run on different switch substrates, such as Linksys wireless router, and to enable them to interact with the other applications. Fig. 1 also shows a generic OpenFlow network hosting two virtual networks, each with its own controller. Moreover, each node hosts an OMNI agent, which communicates with other agents through the knowledge base, which provides a global network view to all agents.

OMNI applications are either passive or active. Passive applications perform network sensing. These applications include flow and switch monitoring, provided by the `Stats` application, and topology discovery, provided by the `Discovery` application. Active applications perform network management tasks. This group of applications includes the `Spanning Tree` application, which computes the network spanning tree to avoid forwarding loops, the `Flow Manager` application, which offers an interface to other applications to add, delete, and modify flows and, finally, the `Flow Migration`, which performs the migration of flows. All of OMNI application resulting outputs are in *eXtensible Markup Language* (XML) to simplify the data interpretation by other applications, by agents, by the graphical interface server, or even by human operators. Finally, the `Web Server` application provides an API that maps web services calls to application functions.

We propose the `Flow Migration` application to migrate flows without losing packets, a functionality not implemented by NOX. The `Flow Manager` application is a simple interface to call functions for creating and removing flows provided by NOX, simplifying the interaction between other applications and the OpenFlow service primitives. The `Stats` and `Web Server` applications are based on a initial code provided with NOX, but we extended their functionalities to monitor other resources and to export applications as web services, respectively.

Finally, the `Discovery` application, provided with NOX, and `Spanning Tree`, obtained at the OpenFlow site, were modified to run on different physical devices such as personal computers and Linksys wireless routers, and to provide their output in XML. Following, we explain in detail the `Flow Migration`, the `Stats`, and *Web Server* applications, as these applications represents OMNI key features.

A. *`Flow Migration`* Application

The main goal of the `Flow Migration` application is to reconfigure a flow that is traversing a particular sequence of switches to another set of switches, after migration. As the network control is centralized and the controller has access to the flow tables of all switches, the flow migration consists in the reconfiguration of flow tables of the concerned switches. The proposed migration algorithm reconfigures the flow tables of the switches in order to prevent packet losses. Indeed, packets are not lost due to flow migration [12] when OpenFlow switches are reconfigured from the farthest switch to the nearest switch, considering the origin to destination flow sense.

The migration algorithm inputs are the current flow description and the sequence of selected switches that the migrating flow should traverse after migration. First, the algorithm identifies the current path of the migrating flow on the network and, then, checks if there is direct connectivity between all switches selected as components of the new flow path. Direct connectivity implies that the next switch of the selected sequence to be added to the new flow path must be a neighbor of the last switch already added to the path. Therefore, the new path is established if the selected switches are a sequence of direct neighbor switches. Otherwise, OMNI computes a path between switches that are not directly connected using Dijkstra's algorithm. Hence, the new path is composed of a

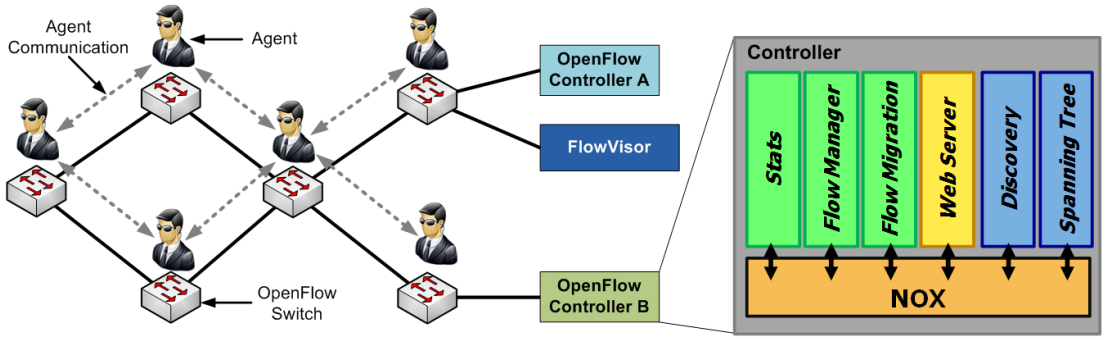


Figure 1. Two virtual networks sharing a physical network, NOX applications, and agent-based system. Each virtual network is controlled by a different instance of the NOX.

sequence of switches, in which the next switch is a neighbor of the current one. Based on the new computed path, the new flow path is added to the switches flow tables in order, from the network output switch to the flow entrance switch. In the flow entrance switch, the controller, updates the previous flow rather than adding a new one. The previous output port is modified to redirect the flow to the new selected path. Thus, the proposed migration algorithm incurs no packet losses. After forwarding the last packet in transit through the original path, OpenFlow deletes the original path because, by default, OpenFlow associates a timer with each flow and deletes unused flow after a time out.

B. `Stats` Application

The `Stats` application obtains statistics about OpenFlow switches and converts them to XML, making them available via a web service. This application sends requests to each OpenFlow switch, which then responds with a report message, containing its description, its statistics, and statistics about each flow of its flow table. This OMNI application is an extension of the *Switchstats* application provided with NOX. The original `Switchstats` application requests switches to provide a description of the switch, the number of packets received and lost by each switch port and also by each switch flow table, but it does not provide any information about flows, because NOX does not implement flow statistic requests[2]. Therefore, we extended the `Switchstats` application as well as NOX to request information about individual instantiated flows and also about aggregated flows, such as a description of each flow, the number of sent packets, the number of lost packets and the duration of each flow. The `Stats` application also calculates the forwarding and loss rates for each flow. Regarding aggregate flows, the application provides the number of packets and bytes sent by all flows in a switch. These additional features are fundamental to manage and to support the flow migration, because this information is input to network-control algorithms, such as the OMNI network-control agent.

C. `Web Server` Application

The OpenFlow controller is a logical centralized entity and all network-control applications run on it. Thus, to enhance OMNI applications interoperability and to enable reusing existing software, e.g. legacy scripts or graphical user-interface programs, OMNI exports a web service interface that interacts with all available applications. The original version of NOX provides a framework for developing a web server that interacts with other NOX applications. Based on this framework, we develop the `Web Server` application as a resource[3] of this framework to provide web services of network control. This application receives function calls as HTTP requests, Processes these requests, and returns the result in XML messages. The XML message returned by each function can be handled and interpreted by other applications, e.g. a user interface applications or an autonomous network control application.

The `Web Server` application interacts directly with other applications via NOX controller internal functions and data structures. Thus, a remote call to the `Web Server` application, encoded as an URL, is mapped into a function call of another application that is running on NOX.

III. Web Interface

The developed web interface for the network monitoring and managing is user-friendly. The web interface is designed as an HTTP client and a server. The client uses services provided by controller applications whereas the server allows remote access to users. Fig. 2 depicts the communication among the OMNI server, the OpenFlow Controller, and the network administrator. The administrator manages the network through the OMNI web interface server. The OMNI interface server translates the administrator operations into requests to the OpenFlow controller that acts on the network.

In OMNI web interface, the network administrator accesses all management features by menus. Facilities that depend on network data update, such as gathering switch statistics, require HTTP requests to the controller application, which processes the request and returns an XML message that is then processed by the interface and, then, generates the visual content. In addition, facilities that modify the network parameters, such as creating flows, also behave as described above, but with the XML response describing the status of the request, which may be a confirmation or an error description.

[2]We use NOX 0.5.0 as reference version of OMNI applications.

[3]Nomenclature used by the framework of NOX.

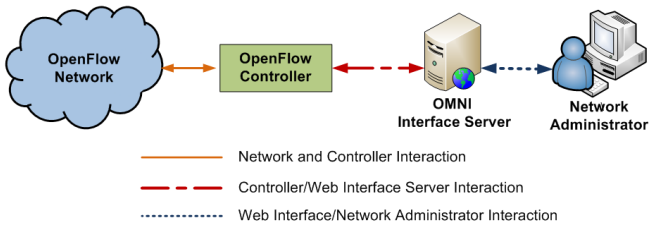


Figure 2. Administrator-Controller communication through the OMNI interface.

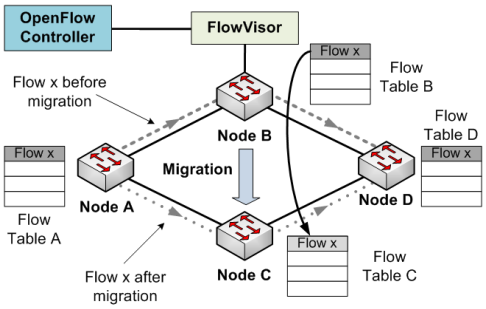


Figure 3. Migration of the flow x from path A-B-D to path A-C-D.

IV. Multi-agent System

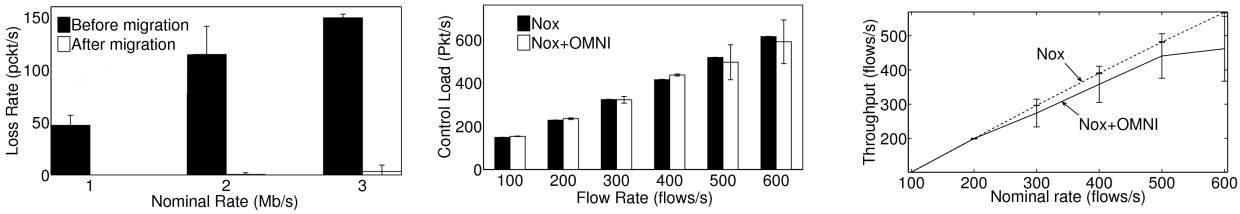
Based on the web service available through NOX and the OMNI Web Server application, we develop an autonomous control mechanism using the Ginkgo agent platform [11]. In order to illustrate this functionality, a multi-agent system is specified to implement failure detection of the forwarding function on a physical node. As soon as an agent detects a failure, the multi-agent system orchestrates the flow migration of all virtual networks that pass through the failing node. The developed agent runs either in a network node or in a middle-box, if a commercial switch does not support the agent. The agent queries the controller about the packet loss rate of the monitored node and stores information in its knowledge base. Therefore, the agent communicates with other network agents, exchanges stored information, and if the agent concludes that a link presents a higher loss rate than a predefined threshold and than other links, it sends a command to the controller, requiring the migration of all flows from the overload link. Furthermore, an agent communicates with all controllers associated to that switch. To do so, it queries the FlowVisor to obtain the controllers that are associated with the switch.

V. Evaluation

We evaluate OMNI to check its response time and its correct operation. We also compare the number of control packets of OMNI and NOX original applications to estimate the OMNI control overhead. We deployed an experimental network using personal computers running Open vSwtich software [13], implementing OpenFlow. Our experimental scenario consists of four OpenFlow switches, a FlowVisor entity and a NOX controller, as shown in Fig. 3. OpenFlow switches and the FlowVisor run on Intel Core 2 Duo computers, with 2 GB of memory. The controller runs on an Intel I7 computer with 4 GB of memory. On this computer, we also run Ginkgo agents, an agent for controlling each OpenFlow switch. We present the results with a 95% confidence interval.

Our first experiment evaluates the migration performed by our multi-agent system. This experiment consists in migrating a flow from the path composed of A, B, and D switches to the path composed of A, C, and D switches, as shown on Fig. 3. The probing traffic is a UDP flow with 1470 bytes of packet size and rate varying from 0.5 to 3 Mb/s. In this scenario, the throughput of the AB link is upper bounded by OpenFlow at 200 kb/s, while the other links, BD, AC, and CD, link are bounded by the link capacity of 100 Mb/s. Therefore, the original path loses packets when the transmission rate exceeds 200 kb/s. As the UDP flow transmission rate varies, we measure the packet losses until the agent autonomously trigger the flow migration. In order to take the decision of a flow migration, the agent verifies the packet loss rate of its monitored switch, which must be higher than 200 packets/s, and also compares the local packet loss rate with loss rates exchanged with other agents. In that comparison, the local value should be at least 20 packets/s higher than the others. The packet loss rate and the comparison thresholds are agent parameters and are set according to each network. Our experiment is an example of agent usage. The developed agent senses the network at fixed intervals of 10 s, and migrates a flow only after three consecutive observations where the packet loss rate is above the threshold. Fig. 4(a) shows that the agent properly detects the bottleneck link within the flow path, and then migrates the flow to avoid, or reduce, the packet loss rate. Since agents observe the loss rate on links at fixed time intervals, the time between starting the agents and the decision of migrating a flow is independent of the packet transmission rate. The agent triggers the migration on average 29.4 s after starting up. Reducing agent-sensing interval, we refine network measurements and reduce agent response time. Reducing this interval, however, also increases the number of statistics messages requested to the controller. Nevertheless, it does not mean an increase in the network control traffic, but an increase in the controller load, as the controller has previously stored information about every switch.

Flow instantiation is one of the main causes of control overhead in an OpenFlow network, because when a packet does not match any flow in a switch, the packet is forwarded to the controller and the controller sends a command to the switch. Thus, our next experiments evaluate the control overhead introduced by OMNI, and the effect of OMNI applications on flow instantiation. We measure the rate of control packets that traverse an OpenFlow network while new flows are instantiated using either NOX or NOX+OMNI. “NOX” means a NOX controller running its original applications for collecting network statistics and for configuring packet forwarding. “NOX+OMNI” represents running all OMNI applications on the NOX controller. Since the NOX+OMNI flow creation mechanism is the same of NOX, all NOX+OMNI control overhead is due to monitor statistics of other resources than NOX. Fig. 4(b) compares the control overhead for a varying



(a) Packet losses before/after running the agents. (b) NOX and OMNI control load. (c) NOX and OMNI flow instantiation rate.

Figure 4. Results of migration experiment calling migration function by an agent and comparison of control overload between NOX and OMNI.

flow instantiation rate. We observe that NOX+OMNI control overhead is negligible for instantiating up to 400 flows/s, as NOX and NOX+OMNI show almost the same control load. When instantiating 500 flows/s and above, the control load of NOX+OMNI is lower than that of NOX, but Fig. 4(c) shows that NOX+OMNI was unable to instantiate as many flows as NOX. The achieved flow instantiation with OMNI reduces and the experiment variation increases, as seen by the error bars. The experiment variation increases because of the test instability caused by the more data that should be handled by controller, e.g. a greater number of packet arrivals as soon as the controller is handling more statistics data about the already instantiated flows. Summing up, comparing NOX and NOX+OMNI shows that OMNI control overhead due to monitoring activity is small. The overhead interferes on network performance when creating more than 400 flows/s, considering our experimental setup. Since OMNI interval for monitoring each resource is configurable, increasing the interval reduces OMNI overhead. Thus, OMNI should achieve higher flow instantiation rate at the cost of increasing the granularity of statistics measures.

VI. Conclusion

The OpenFlow management Infrastructure (OMNI) provides a user-friendly interface to control and manage an OpenFlow network. OMNI is composed of a suite of NOX controller applications, a web-based user interface to access functions provided by the NOX applications, and a multi-agent system to autonomously perform management. OMNI was developed in a modular fashion, thus it can be easily extended to perform new functions.

We implement a testbed to evaluate our OMNI proposal. The multi-agent experiment reveals that our agent is able to eliminate the packet loss after three monitoring intervals. This experiment also shows the correct operation of the flow migration algorithm. When the agent migrates a flow the packet loss rate tends to zero. Another important result is the evaluation of the control overhead of OMNI. Comparing NOX and OMNI+NOX, we conclude that OMNI does not introduce much control overhead, other than statistics about new monitored resources, e.g. statistics about flows. To sum up, OMNI simplifies OpenFlow management and provides the basis for a responsive autonomic control platform while keeping the control overhead almost constant as before.

Acknowledgements

We would like to thank Igor M. Moraes, Marcelo D. D. Moreira, Callebe T. Gomes, Filipe P. B. M. Barretto, Lucas H. Mauricio, and Alessandra Y. Portella for their contribution.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S., and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 69–74, Apr. 2008.

[2] N. C. Fernandes, M. D. D. Moreira, I. M. Moraes, L. H. G. Ferraz, R. S. Couto, H. E. T. Carvalho, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte", "Virtual networks: isolation, performance, and trends," *Annals of Telecommunications*, vol. 66, pp. 339–355, 2011.

[3] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A Distributed Control Platform for Large-scale Production Networks," in *Operating Systems Design and Implementation (OSDI)*, Oct. 2010.

[4] *Simple Network Access Control (SNAC)*. http://www.openflow.org/wp/snac/, Accessed July 2011.

[5] Z. Cai, A. L. Cox, and T. S. E. Ng, "Maestro: A System for Scalable OpenFlow Control," tech. rep., Rice University Technical Report, 2010.

[6] 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.

[7] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.-Y. Huang, P. Kazemian, M. Kobayashi, J. Naous, S. Seetharaman, D. Underhill, T. Yabe, K.-K. Yap, Y. Yiakoumis, H. Zeng, G. Appenzeller, R. Johari, N. McKeown, and G. Parulkar, "Carving Research Slices out of Your Production Networks with OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 129–130, 2010.

[8] K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown, "The Stanford OpenRoads Deployment," in *Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization*, pp. 59–66, ACM, 2009.

[9] R. Wang, D. Butnariu, and J. Rexford, "Openflow-based Server Load Balancing Gone Wild," in *Hot-ICE'11 Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*, pp. 12–18, USENIX Association, 2011.

[10] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S. Lee, and P. Yalagandula, "Automated and Scalable QoS Control for Network Convergence," *Proceedings of the INM/WREN'10*, Apr. 2010.

[11] M. Abid, I. Fejjari, and G. Pujolle, "An Autonomic Piloting Plane for the Handover Decision Optimization," in *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on*, pp. 1 –5, Dec. 2009.

[12] P. S. Pisa, N. C. Fernandes, H. E. T. Carvalho, M. D. D. Moreira, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte, "Openflow and Xen-based Virtual Network Migration," in *Communications: Wireless in Developing Countries and Networks of the Future* (A. Pont, G. Pujolle, and S. Raghavan, eds.), vol. 327 of *IFIP Advances in Information and Communication Technology*, pp. 170–181, Springer Boston, 2010.

[13] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending networking into the virtualization layer," in *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking* (ACM, ed.), Oct. 2009.