

SLAPv: A Service Level Agreement Enforcer for Virtual Networks

Hugo E. T. Carvalho, Natalia C. Fernandes, and Otto Carlos M. B. Duarte
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro –Brazil
{hugo, natalia, otto}@gta.ufrj.br

Guy Pujolle
UPMC Sorbonne Universités
Paris – France
{Guy.Pujolle}@lip6.fr

Abstract—In this paper we propose SLAPv, an efficient control system for service level agreements (SLAs) in virtual network environments. SLAPv fills a gap in the design of SLA enforcers by introducing the following set of control features: (i) verification of physical resource usage, (ii) retrieval of real-time profiles of virtual routers, and (iii) enforcement of the adequate usage of resources. Furthermore, SLAPv punishes virtual networks that do not respect their contracts. The control relies on fuzzy logic to enforce resource allocation according to usage profile of routers and the system load. The punishment depends on the exceeding violating value, and on the system load. Results obtained from a developed prototype show the efficiency of the proposed system under different conditions and strategic policies. Tests performed with RIPv2 running on virtual networks show that, for high load scenarios, misbehaving routers are forced to rapidly enter in conformity to their SLAs in less than five seconds.

Index Terms—Virtualization, QoS, Fuzzy, Future Internet

I. INTRODUCTION

The resource virtualization techniques allow the execution of multiple operational systems over the same physical hardware. This functionality is performed through a monitor, which is responsible for multiplexing hardware access and providing logical resource slices to the virtualized systems. The virtualization, which is based on the premise that virtual environments are isolated, offers advantages such as better usage of physical resources and the ease to dynamically remap virtualized resources among virtualized systems. One of the most used open source virtualization platforms is Xen [1]. Xen architecture allows innovation in computer networks because each virtual machine can act as a virtual router. Thus, a single physical router can support multiple virtual routers with distinct properties and protocol stacks, which is suitable for the pluralist architecture for the Future Internet [2].

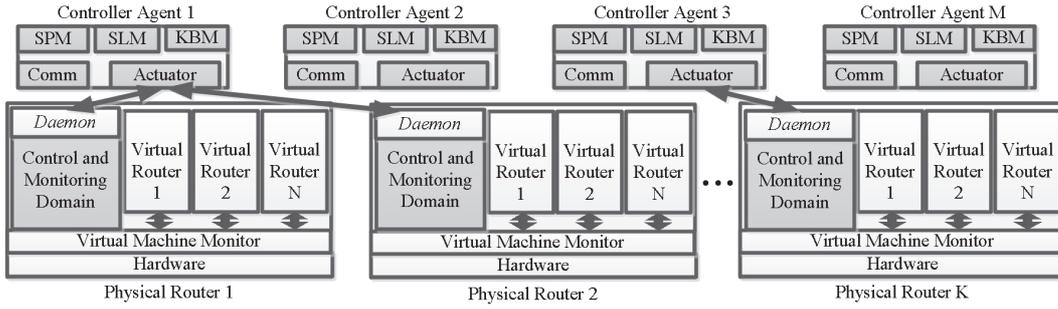
In this paper we propose a dynamic resource controller to enforce service level agreements (SLA) in virtual networks, called SLAPv: service level agreement policies in virtualization. The originality behind SLAPv is that it relies on fuzzy logic to account for enforcing the proper behavior of virtual routers and their conformity with SLAs. The control relies on the generation and analysis of usage profiles of each virtual router. Based on this information, SLAPv detects service level violations. Misbehaving networks that violate contracts are promptly punished. Besides, the control system monitors the load of each virtual network and estimates usage profiles of each router. These router profiles are used to guarantee a better distribution of resources on physical routers, reducing

the occurrence of overloads in physical routers. SLAPv also enables flexible enforcement. The proposed SLA enforcement depends on the physical system load, which means that under scenarios with plenty of resources available, SLAPv allows virtual networks to violate SLAs for some time. This flexible enforcement increases the resources utilization of the whole system and enhances the quantity of resources offered to virtual networks. Fuzzy logic is used to map flexible administrator strategies onto punishment rules, which are applied to misbehaving virtual routers. In this context, the proposal allows the control of SLAs with different strategy patterns. It also allows administrators to easily insert/modify/remove acting strategies.

SLAPv achieves resource control by relying on a long list of load parameters. Besides CPU, memory, and network usage (traditionally used in the literature [3]), SLAPv also considers robustness and temperature of operation. These parameters are useful to select less error prone physical machines to avoid the allocation of critical virtual machines in unstable physical routers. SLAPv was implemented and tested. Results show that the proposal correctly punishes router violations and that the punishment is adaptive, depending on the availability of physical resources. The system conforms misbehaving routers in less than five seconds for scenarios with scarce resources. On the other hand, SLAPv reacts smoothly for scenarios with abundant resources, allowing routers that violate SLAs to obtain an additional amount of resources. The article is structured as follows. Section II describes related works and Section III explains the system elements, the generation of usage profiles, the flexible strategy system and the load and punishment controllers. Section IV shows the system results, which demonstrate the performance of the controller under different environment configurations. Section V presents our conclusions and future directions.

II. RELATED WORK

Most of the works on dynamic allocation of physical resources in virtualized systems focus on data center server consolidation [3]–[5]. Sandpiper [3] is a system that monitors virtual machines in data centers automatically accomplishes the migration of machines. The objective of Sandpiper is to optimize the performance of each server and ensure proper working of virtual machines when they suffer behavior changes, such as an abrupt augment of accesses due to a flash crowd. Sandpiper monitors profiles of virtual machines



SPM – Strategy and Policy Module SLM – Service Level Module KBM – Knowledge Base Module Actuator – Interaction with Daemons Comm – Communication with Agents

Figure 1: SLAPv architecture. Actuator of the controller agents interacts with daemons of physical routers.

through time series and estimates the volume of load based on CPU, memory, and network usage.

Xu *et al.* propose two-level control mechanisms for data centers, where the first one is global and maps virtual resources in physical resources and the second is devoted to local control [5]. The local controller maps the relation among workloads and resource usage to predict future workloads of each virtual network. Both controller functions are based on fuzzy logic. Menascé *et al.* apply autonomic computing techniques to control the share of processors among virtual machines in data centers [6]. Authors use optimization techniques to maximize a global utilization function on the physical nodes and apply this information in a white box controller.

Keller *et al.* evaluate quality of service requisites in virtualized environments and propose authoring models to make the accountability and ensure SLAs in virtual routers [7]. Authors suggest two models to guarantee authoring and QoS: a model based on the monitoring of network parameters to analyze service levels and other based on trusted computing platforms.

There are also platform-specific proposals to solve resource distribution in virtualized systems, such as Xen. Fernandes and Duarte proposes *Xen Network Monitor*, which implements a secure control system to restrict I/O resource usage among virtual routers [8]. These systems controls specifically the resources shared in a privileged domain which is shared among all virtual networks in network operations, depending on pre-defined SLAs.

The proposed system is a local controller to virtual network environments independent of virtualization platform. Although the proposed mechanism can be used in data centers, the developed logic aims at solving challenges in virtual networks environments. This scenario differs from those found in data centers, because the main controlled parameter is not limited to processor usage, but also considers parameters that are needed to control packet forwarding, which depends of the traffic, memory and so on.

III. THE PROPOSED CONTROL SYSTEM

SLAPv: service level agreement policies in virtualization aims at monitoring and guaranteeing service level agreements (SLAs) in virtual networks. SLAs are defined as contracts established between infrastructure service providers and their customers, which regulate parameters such as bandwidth, delays, consumption of resources, guarantees and fines. The

key idea is the generation and analysis of use profiles of virtual routers and the punishment of misbehaving routers, using fuzzy logic controller.

SLAPv follows a distributed management model composed of controller agents. Each agent is associated with a set of physical routers where virtual routers execute, as seen in Figure 1. Each controller agent can be associated to a given number of domains and the network manager must decide on how these associations are accomplished. Each physical router has a control domain where a daemon monitors the allocation of physical resources, verifies in real time the conformity of SLAs, and generates the use profiles of each virtual router.

Five modules compose the controller agents. The Strategy and Policy Module (SPM) holds management strategies that can be applied over physical routers under its control and updates the current strategy to be applied on each control and monitoring daemon. The Service Level Module (SLM) keeps a database that associates the SLAs with each virtual router. The Knowledge Base Module (KBM) stores the history, use profiles and the description of violations that have occurred. This module can be used to estimate future network migrations, detect the need for updates and re-negotiate contracts, adapting them to the real profile of each virtual router. The Actuator Module executes within the control and monitoring daemons and retrieves the profiles and statistics of each virtual router. The controllers also have a Communication Module (Comm), which allows the exchange of information among other controllers through secure channels. If necessary, the controllers may use those channels to negotiate actuation domains changes and negotiate virtual elements migration.

Physical routers also participate from the control system through the daemon that executes within the monitoring and control domain. This daemon, besides receiving data from the controller agents, such as the level of punishment for each virtual router, also performs two important functions. First, the daemon evaluates and generates resource patterns to create the virtual router profiles. Besides that, the daemon estimates the system load based on monitored data for each virtual router.

A. Generating Router Profiles

The use profile of each virtual router represents the resource consumption pattern of each virtual router. Profiles can be used to detect rule violations, to estimate the use of resources, and to predict future needs. Profiles are generated through the capture

of memory, processor and network utilization over time, to store the recent past and the long past of those variables. There are two sliding windows with distinct sizes to store those two time series. The sliding windows help the detection of time correlation between measurements. The recent past reflects a closer machine state and allows fast detection for SLA violations whilst the long past is used to predict future behaviors and estimate possible future demands. The generation of profiles based on probabilistic density functions is used in Sandpiper [3] to capture time series which reflects the resource consumption pattern of memory, processor and network. Behaviors are analyzed through probability functions. In our proposal, we use probability density functions (PDFs) to demonstrate the probability that a given event happens inside a set of measurements in the long past window. So, the PDF can indicate, among others, the probability of consumption of resources, such as processor use. We also use Cumulative Distribution Functions (CDF) to verify service level agreements in the short past window. The CDFs describe the amount of resources that were utilized in relation to the total time inside the window. Thus, it is a good estimator to define and verify fulfillment of SLAs. Through this perspective, we can think of flexible SLAs.

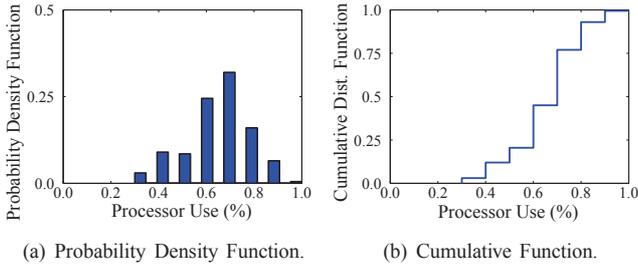


Figure 2: Processor utilization profiles of a RIPv2 virtual router.

In order to exemplify the profile use based on probability distributions, we have developed a prototype to generate profiles and generated a test¹, as seen in Fig. 2. A probability distribution function of processor use inside a virtual router with RIPv2, generated by our system, can be seen in Fig. 2(a). We can see that the exchange of control messages generated an approximated use of 0,7% of CPU in 30% of the measurements in the long past window observed. The cumulative distribution function of the same scenario can be seen in Fig. 2(b). There, we can see that the virtual router uses, in average, only 0,7% of processor resources in 80% of the observations in the long past window. Due to this properties, we can conclude that in this scenario we can aggregate a small set of RIPv2 routers with similar properties (for example, 10 routers) and allocate a single processor to be shared among them, avoiding performance loss in the exchange of control messages.

¹In this test scenario, there are four physical routers executing Xen and inside each routers there is a virtual router. We create logical links among them and execute RIPv2 routing protocol with XORP [9], which exchanges control messages. We have also considered a long past window of 200 observations to generate the PDF and the short past window of 20 observations to generate the CDF. The sampling interval is one second between measurements.

B. Estimating System Load in Control Daemons

The system load is a measurement that indicates the load level of the managed hardware. Thus, it is possible to detect saturation of resources. The load measurement involves multiple parameters such as processor use, memory use, network use, temperature of components, system robustness and so on. The monitoring uses information gathered from the operational system from the monitoring and control domain. The monitored information is processed by a fuzzy controller to generate the system load.

We have adopted the fuzzy controllers due to its adequacy for decision making problems that involve uncertainties, imprecisions or qualitative values, such as the strategies of a network manager. In fuzzy logic a given element belongs to a group according to a membership level inside the continuous interval $[0, 1]$, where $\mu_A(x) : X \rightarrow [0, 1]$ defines a membership function. So, given the set of membership functions $\mu_{Proc}, \mu_{Mem}, \mu_{Net}, \mu_{Temp} \in \mu_{Rob}$, corresponding to processor, memory, network, processor temperature and system robustness, respectively, we can associate each of the resources or parameters into fuzzyfied variables. It is important to notice that the specified curves, which are called membership functions, reflect the configuration of a particular network manager.

The combination of membership levels generates an output defined as system load, which has values in the interval $[0, 1]$ and represents the load of physical resources of the system. This value can then be used to estimate future migrations. Besides, the system load is used as input parameter to the punishment controller. The punishment controller also receives the input delta, which is used to estimate the level of the violation performed by virtual routers. A block diagram of the Strategy and Policy Module (SPM), which has the two mentioned controllers², can be seen in Fig. 3.

1) *Strategies Based on Inference Rules:* The fuzzy controller strategies are based on the default fuzzy inference rules of the fuzzy system. Those rules follows the IF \rightarrow THEN pattern which represents the current action strategy scheme. The set of rules that defines a strategy is named a strategy package. An example of strategy package that calculates the punishment level according to the difference between the contracted SLA and the current resource use, denoted by delta, and the system load can be seen in Table I.

Strategy Package
If Delta (low) and Load (low) Then Punishment (low)
If Delta (average) and Load (low) Then Punishment (low)
If Delta (high) and Load (low) Then Punishment (average)
If Delta (low) and Load (high) Then Punishment (average)
If Delta (average) and Load (high) Then Punishment (high)
If Delta (high) and Load (high) Then Punishment (high)

Table I: Example of a piece of a strategy package.

The presented strategy package corresponds to a network manager policy that establishes that even huge SLA violations is not punished severely, when the system is lightly loaded, because the system has plenty of resources and at this moment,

²To both controllers, we adopt the Mamdani implication method, with Zadeh [10] logical operations and the centroid method of defuzzification.

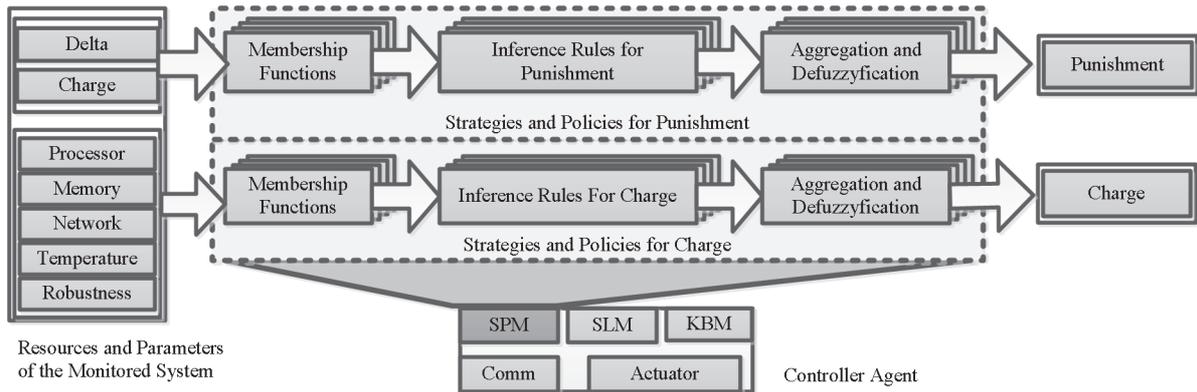


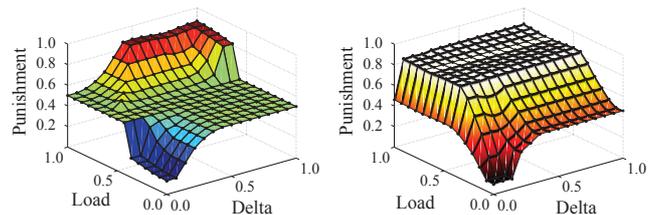
Figure 3: Strategy and Policy Module.

giving additional resources to the violating router do not disturb the others. On the other hand, when the system is overloaded, the network manager is more rigorous and even light violations are punished severely. These rules work together with the membership functions that can be also developed by the network manager. It is possible to easily insert new rules and strategies and the controller agent must export the strategy packages to the daemon that must use the defined strategy. We can also establish different strategies for each resource under control, enhancing the flexibility of the controller.

2) *Load Policies*: After the application of the inference rules, fuzzy values are generated to represent the level of pertinence of each inference rule and then those values are mapped into a single controller output, which is a value in the interval $[0, 1]$ which represents the current system load.

Given the configurations and membership functions, it is possible to verify the relationship among the punishment level, the system load and delta, i.e., the level of SLA violation. This relationship can be viewed in Figure 4. In those surfaces, we can see that a given policy defines how the punishment level varies. Besides, the combination of different inference rules and membership functions generate different surfaces. In the conservative policy, we can see that punishments are severe only when delta is high and the system is overloaded (Figure 4(a)) whilst in the aggressive policy, small positive variations in system load and in delta generate high levels of punishment (Figure 4(b)). The application of a conservative strategy may allow a better usage of idle resources by violating routers, while the aggressive strategy can let idle resources with the tradeoff of ensuring that all virtual routers can use its own share of resources and that the system will rapidly control violations in SLAs.

3) *Controlling the System Overload and SLAs*: SLAPv is capable of generating use profiles, evaluating if profiles correspond to established SLAs, generating estimating the system load and punishing virtual routers that violate the proposed rules. In the implementation of the prototype, the daemon that executes within each domain performs the parameter gathering at each given sampling interval, which can be defined by the network manager. The parameters are used to generate temporal series that represents the variation in the use of each resource and the statistics and distributions that allow



(a) Conservative decision surface.

(b) Aggressive decision surface.

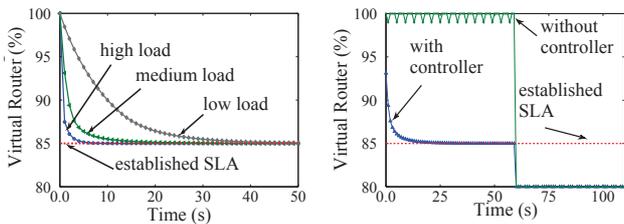
Figure 4: Decision surfaces for different management strategies.

the verification of profiles and the compliance with SLAs. This information is sent to the controller agent. The daemon within the controller agent verifies if the use profiles of virtual routers match the contracted SLAs. Besides, it aggregates the resources used by each virtual routers to estimate the total physical system load. Based on this information, we can calculate delta. The system then uses delta and the system load to make the decision of which is the adequate level of punishment that must be applied in the violating router.

IV. RESULTS

In order to validate SLAPv, we developed a few experiments with a prototype which proves that the proposal works and adaptively control SLAs. In the test environment, we selected the Xen platform to create virtual routers. The control parameter used in the tests is the caps. In Xen, caps regulates the amount of processor use that each virtual element can use. By doing so, when we manipulate caps, we can control the processor resource use of each virtual router. There are also other tools that can be used to control the network, such as Traffic Control (TC) [11], which allows the management of packet queues, managing the throughput of each virtual router in case of SLA violations. Tests were performed in a physical machine with a core i7 860 processor with four real cores and 8 GB RAM DDR3. The machine is configured with Xen hypervisor 4.0. Virtual routers are configured with 128 MB of RAM memory and access to a single virtual processor. Both virtual routers and monitoring and control domain are configured with Debian Lenny and kernel 2.6.32-5-amd64 with the needed patches to enable support to Xen.

The first experiment verifies the behavior of the proposed controller and the fuzzy punishment mechanism. In order to execute the test, we select one virtual router. The SLA of this router defines among other rules that the router can use up to 85% of processor to execute packet forwarding and exchange of control messages. After that, we have created a packet flow that is forwarded through the router. When the virtual router forwards the traffic, a SLA violation occurs. The router extrapolates its SLA and the control system regulates through caps the amount of processor that can be used by the virtual router. In the experiment, the interval between SLA verifications was defined as one second. The punishment system was enabled 60 seconds after the time when the router started using more than its SLA allows.



(a) Different system loads. (b) Router violating SLA until $t=60$ s.

Figure 5: Efficiency of the controller in different scenarios.

To perform these experiments, we define three scenarios. In the first scenario, there is only the monitored router itself and other virtual router that only exchanges control messages from RIPv2 with other routers in other machines. In this case, the system load metric was considered low. In the second scenario, there is the monitored router and a set of five virtual routers, each one using a moderate amount of resources, which generated a load value of approximately 0.5, defined in the system load metric as medium load. In the third scenario, there are seven virtual routers beside the monitored one and all of them are using resources up to their SLA limits. The system load was reported as approximately 0.6, which was defined as high system load. For each of the scenarios we execute only a single test round. Results seen in Fig. 5(a) shows that the system converges to ensure the established SLA. In these tests, we used the conservative policy. Depending on the system load, for each environment, the punishment level varies. We can see that in low load scenarios, the punishment level is low and the system takes up to 40 seconds before the SLA is fully respected. In medium load, the punishment is more intense and converges in less than 15 seconds. In the high load scenario the punishment is severe and the system converges to the SLA in less than five seconds. We realize that our proposal fulfill the established requisites, acting in a conservative fashion in situations with plenty of resources and aggressively in critical situations. If significant changes happens in the system load value, the observed results will change, reflecting the system load variation during the convergence of the system.

The second experiment validates the proposed controller when a given virtual router extrapolates its SLAs for a given period, and after that, respects its SLAs. In the experiment, the virtual router forwards a huge amount of traffic and saturates its processor. After 60 seconds, the router forwards at a lower

rate, respecting its SLA due to the processor usage of 80%. In this result, the system load remained as medium. Results are shown in Fig. 5(b). We can observe that without the controller, the virtual router can use all the available resources, allowing it to hassle the behavior and SLA of other virtual routers. We can see that in this case the processor measurement shows some variations. This happens due to small oscillations in the measurement tools. When the caps parameter limits a router to use less than it tries to use, this limit is due to the caps so there is no variations. When the virtual router uses an amount of processor that is smaller than caps, variations can occur. When we use SLAPv, the virtual router has its caps limited gradually until it starts to respect the established contract.

V. CONCLUSIONS

In this work, we developed SLAPv, a fuzzy control prototype to enforce service level agreements in virtual network environments, in which the lack of isolation properties represents a challenge. Network managers can easily design and apply their own strategy packages that is a set of rules that reflect their own experiences in network decision making. The results show that the system can control the established SLAs, punishing routers that violate SLA rules, according to the system load and the violation level. In the test environment, the system limited adaptively the SLA. When there are plenty of resources, the SLAPv applies light punishments and in critical moments the system applies severe punishments. The system rapidly converges in less than five seconds for heavy load scenarios and, on the other hand, smoothly accommodates the resource usage in up to 40 seconds when resource load is low in our test scenarios.

REFERENCES

- [1] M. Bourguiba, K. Haddadou, and G. Pujolle, "Evaluating xen-based virtual routers performance," *International Journal of Communication Networks and Distributed Systems*, vol. 6, no. 3, pp. 268–282, 2011.
- [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," vol. 66, no. 5-6, pp. 339–355, Jun. 2011.
- [3] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the Networked Systems Design and Implementation (NSDI)*, 2007, pp. 229–242.
- [4] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of INFOCOM 2010*. IEEE, 2010, pp. 1–9.
- [5] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches," *Cluster Computing*, vol. 11, no. 3, pp. 213–227, 2008.
- [6] D. Menascé and M. Bannani, "Autonomic virtualized environments," in *Autonomic and Autonomous Systems, 2006. ICAS'06. 2006 International Conference on*. IEEE, 2006, pp. 28–37.
- [7] E. Keller, R. Lee, and J. Rexford, "Accountability in hosted virtual networks," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*. ACM, 2009, pp. 29–36.
- [8] N. C. Fernandes and O. C. M. B. Duarte, "Xnetmon: A network monitor for securing virtual networks," in *Proceedings of IEEE International Conference on Communications (ICC'11)*, Jun. 2011, pp. 1–5.
- [9] M. Handley, O. Hodson, and E. Kohler, "XORP: An open platform for network research," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 53–57, 2003.
- [10] L. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.
- [11] W. Almesberger *et al.*, "Linux network traffic control: implementation overview," in *Sixth IEEE Symposium on*, vol. 2001, 1999, pp. 296–301.