VNEXT: Virtual NEtwork management for Xen-based Testbeds

Pedro S. Pisa[1], Rodrigo S. Couto[1], Hugo E. T. Carvalho[1], Daniel J. S. Neto[1], Natalia C. Fernandes[1], Miguel Elias M. Campista[1], Luís Henrique M. K. Costa[1], Otto Carlos M. B. Duarte[1], and Guy Pujolle[2]

[1]Universidade Federal do Rio de Janeiro, Rio de Janeiro – Brazil
[2]UPMC Sorbonne Universités, Paris – France
Emails: {pisa, souza, hugo, neto, natalia, miguel, luish, otto}@gta.ufrj.br, and Guy.Pujolle@lip6.fr.

***Abstract*—Network testbeds strongly rely on virtualization that allows the simultaneous execution of multiple protocol stacks but also increases the management and control tasks. This paper presents a system to control and manage virtual networks based on the Xen platform. The goal of the proposed system is to assist network administrators to perform decision making in this challenging virtualized environment. The system management and control tasks consist of defining virtual networks, turning on, turning off, migrating virtual routers, and monitoring the virtual networks within few mouse clicks thanks to a user-friendly graphical interface. The administrator can also perform high-level decisions, such as redefining the virtual network topology by using the plane-separation and loss-free live migration functionality, or saving energy by shutting down physical routers. Our performance tests assure the system has low response time; for instance, less than 3 minutes to create 4-node virtual networks.**

I. Introduction

The employment of virtualization techniques in network scenario allows the execution of multiple virtual networks, called slices, over the same physical substrate [10]. Virtual networks are isolated and each one can use its own protocol stack. Thus, network virtualization enables providing a pluralist Internet where different virtual networks coexist instead of a single and generic network. Because virtualization offers the required flexibility for the future Internet and also allows TCP/IP-stack protocols to run in the slices, it has been used as the basis for network testbed platforms [7], [14].

Virtual networks are implemented with the same virtualization approach used for server consolidation. Similarly, each physical router is shared among multiple virtual routers. A virtual network is then the set of virtual routers and their respective virtual links [8], [17]. Router virtualization can be accomplished with Xen [1], which is an open source virtualization platform for commodity computers. Egi *et al.* show that current multicore personal computers can represent a low-cost alternative for commodity routers with similar performance [8]. In addition, whereas commodity routers have only one control plane and one data plane, Xen enables the virtualization of both planes. This characteristic allows each virtual router to have its own control and data plane, enhancing network programmability. This is the major advantage of using Xen in network virtualization, used in different Future Internet testbed facilities and new Internet architectures [7], [14].

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

The tradeoff of network virtualization is the management overhead, which becomes a challenge even harder than in the current Internet [16]. Allocating physical resources among virtual networks and also managing and controlling the resources used by the multiple virtual networks are complex tasks [9], [12]. In such scenario, network administrators require the assistance of management and control systems powered by decision making features. The number of monitored network parameters and the influence of a virtual network onto the performance of all the other virtual networks considerably increase the importance of these systems. Besides, there are new functionalities which could represent additional possibilities not yet integrated in virtual network management systems, such as instantiation, migration, and shutting down of a single virtual router [11] or an entire virtual network.

This paper proposes VNEXT (Virtual NEtwork management for Xen-based Testbeds), a management and control system for virtual networks based on the Xen platform. Existing platforms are often proposed for data centers with functionalities such as instantiation, shutting down, and migration of virtual routers. VNEXT is specific for managing and controlling virtual routers and networks, and therefore, is different from existing management systems also based on Xen. HyperVM [13] is designed to manage a virtualized server cluster, providing a web based graphic interface and other tools like traffic shaping and network management. In spite of using management tools for network virtualized environment, the HyperVM is not suited for network virtualization because it does not consider the virtual machine acting as a router. Consequently, this system does not provide specific functions as topology management and plane separation. On the other hand, the MLN (Manage Large Networks) [2] system can manage a virtual network environment using Xen and other virtualization technologies such as UML (User Mode Linux). MLN defines a language to build and manage a virtual network, but it does not consider performance issues like monitoring infrastructure resource utilization or providing plane separation as VNEXT. Moreover, VNEXT yields virtual router live migration without packet loss and offers a tridimensional interface for visualization of the network. With the simple and friendly VNEXT user interface,

it is possible to make high level decisions like redefining virtual network topology, for instance, saving energy through shutting down or sleeping physical routers.

This paper is organized as follows. Sections II and III describe the architecture of the VNEXT system and its main functionalities. Section IV evaluates the performance of the system and Section V concludes this work and presents our next steps in VNEXT development.

II. VNEXT Architecture

The VNEXT architecture is composed by three main components: a virtual network controller, *daemons* for monitoring and acting on physical and virtual routers, and a graphical interface for network administrators. The controller exports control and management primitives, which are triggered by the network administrator through the graphical interface, and interacts with the *daemons* executed in each router for implementing user decisions. Figure 1 presents the VNEXT functional architecture.

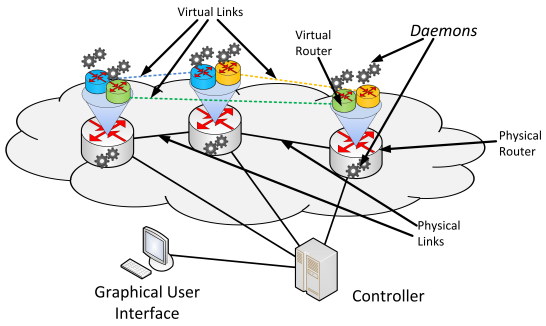


Figure 1. VNEXT architecture: controller, monitoring and acting *daemons*, and graphical user interface.

The monitoring and acting *daemon* are the main component of the system. Every physical and virtual router in the network executes the *daemon*, which basic functionalities are: collecting information requested by the controller, implementing administrator decisions in the routers, and formatting the information obtained from the different routers according to a pre-established standard. The collected information comprises memory usage, bandwidth consumption, number of processors, and percentage of the system or application processor usage. All these information are obtained by passive and active measurements using tools specifically developed for VNEXT, i.e. designed for virtual network environments, and also by using existing tools, such as `ifconfig`, `top`, and `iptables`. In addition to monitoring, *daemons* are also responsible for acting in physical and virtual routers, implementing controller requests as is described in Section III. For instance, daemons are in charge of migrating or shutting down virtual routers upon controller requests.

The communication among the *daemons* is hierarchically organized to reduce network control traffic. The controller only communicates with physical router *daemons* and the physical router *daemons* communicate with the virtual router *daemons*. This hierarchy improves isolation because virtual routers are not in the controller network.

The *controller* component of VNEXT intermediates the communication among users and router *daemons*. The controller receives network administrator requests through the graphical interface and sends requests to the routers. It is worth mentioning that any other interface or client software provided by the administrator could also be used instead of our proposed graphical interface. The controller runs in a separated commodity computer or in any physical router of the network. The controller is implemented in Java and the services are exported as Web Services, using the Axis2 library and the Apache Tomcat6 server. All communication between the controller and the *daemons* are performed by TCP sockets and XML (*eXtensible Markup Language*) messages using a standard XML header. The controller also exports a control service interface used by the graphical interface. The service interface uses SOAP (*Simple Object Access Protocol*) to encapsulate XML messages and to provide the features described in Section III. The controller also stores the current state of the physical and the virtual routers, keeping, for instance, a list of available physical routers and their respective virtual routers. The activity state of the physical routers is monitored through keep alive messages periodically sent from the physical router to the controller. Virtual router states, however, are maintained via `Libvirt` library requests, executed in the physical routers to inform the virtual routers state.

The *graphical user interface* (GUI) component provides a friendly control and visualization of the physical and virtual networks for the administrator, as shown in Figure 2. It was developed in Python with Qt and OpenGL libraries. The GUI communicates with the controller using a wrapper, which encapsulates controller SOAP services in Python methods. In the VNEXT GUI, the physical and virtual networks are structured and exhibited as graphs in a frame that provides rotation, translation, and zoom interactions. The graphical interface includes algorithms for reorganizing the topology and for calculating the shortest path, based on number of hops or on link weights, computed according to link latency. The GUI also shows the current available physical and virtual routers, the collected router monitoring information, and the current networks running over the infrastructure. Regarding network actuation, the GUI provides dialogs for turning on and off physical and virtual routers, and rebooting, suspending, resuming, and migrating virtual routers. Users create new virtual networks drawing it in the topology visualization window. The operations are performed in batch when the user submits the new network to the controller.

III. VNEXT Features

In this section, we describe VNEXT main features. Detailed information is available at http://www.gta.ufrj.br/vnext.

Virtual router creation: A virtual network is composed of virtual routers and the virtual links connecting them. In the VNEXT context, a virtual router is a Xen virtual machine executing routing software. Therefore, the network admin-

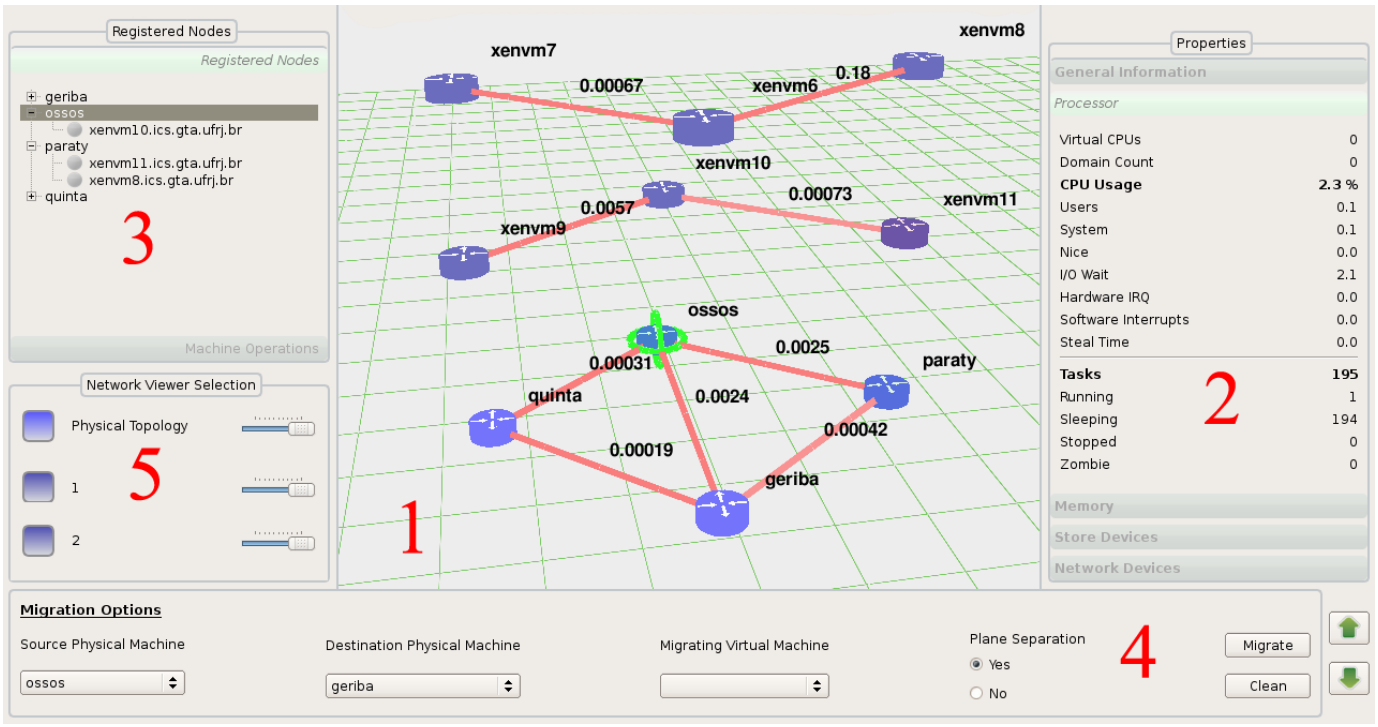


Figure 2. (1) network topology, (2) router information, (3) router availability information, (4) migration mechanisms, and (5) customization settings.

istrator defines the virtual router requirements in terms of memory, virtual CPUs, and network interfaces. Virtual network parameters are used for assuring data plane isolation when data plane are inside the virtual router [6] or sharing the Domain 0 [10]. VNEXT creates a new router by cloning a pre-defined router disk image in order to avoid real time operating system installation. Besides, after cloning the router image, VNEXT accesses the new router file system to install additional routing features and configure specific files for each virtual router, e.g. network interfaces and routing protocol configuration files. These facilities provide administrator access and communication among the new virtual routers because it defines address and initial routes. Once the virtual routers are created, they can be instantiated at any time.

Virtual network creation and individual network control: Creating virtual networks on demand is an important facility required for network operators [5]. After creating virtual routers, the next step is to map the virtual routers into distinct physical routers and interconnected them through virtual links. Therefore, creating a virtual network means configuring the virtual router interconnections, setting the network IP address, and defining the routing protocol to be used. The network administrator uses the graphical interface to view the physical topology and then to create a new virtual network. The administrator defines virtual router requirements and the virtual network topology by drawing the virtual routers and virtual links. The new virtual network is submitted as batch operation, which can be executed in serial or in parallel, for creating each virtual router. After creating the whole virtual network, it can be instantiated instantaneously, i.e. the virtual network routers can be turned on. VNEXT can also shut down or delete virtual networks to immediately release physical resources.

Monitoring routers and links: Monitoring network elements permits the network administrator to anticipate future network behaviors [18] and to evaluate the utilization of the network resources [4]. The developed system provides real-time information related to virtual routers and links. The GUI main window exhibits the physical and virtual network topologies, presenting the latency of links in milliseconds. When the user selects any router, the information about the selected router is collected by the monitoring *daemons*. Among the information, there are the current latency of the physical links, the rate and the number of transmitted packets, as well as the resource usage, such as memory and CPU.

Virtual topology rearrangement: Network virtualization allows live migration of a virtual router from one physical router to another. This is possible only if all the physical routers run the same virtualization platform. The migration advantage over a simple virtual router copy is the maintenance of the routing service without disruptions. VNEXT implements plane separation migration [15], which migrates a virtual router without any packet loss. This is possible because the virtual router forwarding plane is kept functional during all migration process. Nevertheless, VNEXT also implements the standard Xen migration, which, on the other hand, may lose packets.

Turning on/off physical routers: Proposals in the literature consider sleeping and turning off network routers to save energy [3]. VNEXT has mechanisms to turn on/off physical routers in order to execute preemptive maintenance or save energy in case of failures or low network load. When the user requests a physical router to be shut down, all of its virtual routers are first migrated to another physical router. In the GUI, the network administrator chooses the physical router to turn off and the router that will receive the virtual routers.

The shutting down process is performed through a command to the daemon that turn off the physical router. On the other hand, to remotely turn on a physical router, VNEXT uses the Wake-on-Lan mechanism of Ethernet.

IV. Evaluation

We conduct an experimental evaluation of the VNEXT system in a testbed . Our goal is to analyze the scalability of VNEXT for an increasing number of virtual routers in the network. We measure the time needed to create a virtual network and to collect data from physical and virtual routers. In addition, we obtain the response time of important VNEXT functionalities, such as physical routers turning on and off.

A. Testbed setup

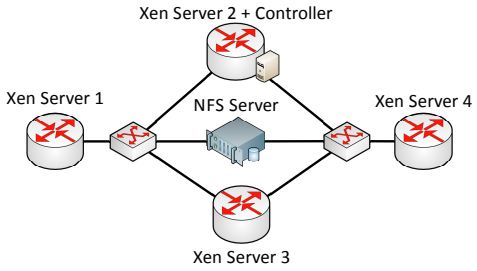


Figure 3. The testbed topology. All servers are connected through switches and can access the NFS server.

Our testbed is composed of five physical machines running Linux Debian operating system. All machines are equipped with Intel multicore i7 processors and 16 GB 1333 MHz DDR3 of RAM. The testbed topology is seen in Figure 3. The Xen Servers execute Xen 4.0 virtualization technology and the Xen Server 2 also plays the role of VNEXT controller. The NFS Server stores all the disks of virtual routers and is accessible from all Xen Servers, which must use the NFS server to instantiate their virtual machines. It is important to note that the NFS Server is a disk server configured with RAID-5 to enhance reliability and performance. Finally, all physical Xen Servers and all virtual machines execute VNEXT *daemons*.

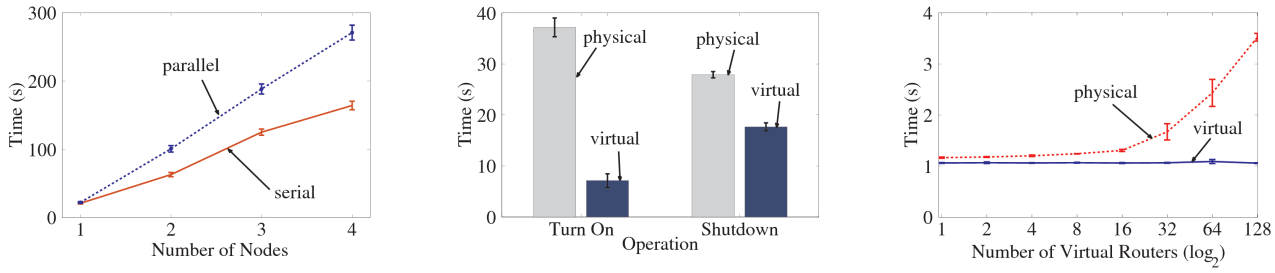
B. Results

The first experiment measures the time required to create a virtual network according to an increasing number of virtual routers. The virtual network creation procedure is executed by VNEXT controller, but the disk image copy and configuration of each virtual router are executed on the NFS Server. After creating the virtual routers, each one is instantiated on its respective physical machine. Figure 4(a) shows our results for network creation using serial and parallel approaches. In the serial approach, the NFS Server creates one by one virtual router. In parallel approach, the NFS Server instantiates a thread for each virtual router creation, thus it starts the creation of them simultaneously. Results show that the serial approach performs better than the parallel because the NFS Server becomes a bottleneck. The copy of the virtual disk images cannot be executed at the same time on the disk. Therefore, the concurrent threads contend for the disk access for writing. This contention leads to a decrease in performance because the scheduling of the multiple writing threads must deal with the overhead for disk seeks. Each virtual router image is written on different disk sectors, for instance. In the serial approach, on the other hand, writing on disk is continuous and the use of functionalities such as disk burst write further reduce the duration of the disk image copy. As the time required to perform disk image copy is the main component of the whole process of virtual network creation, we decided to choose the serial approach in VNEXT implementation. This choice allows the creation of a 4-node virtual network in less than 3 minutes.

The process of instantiating virtual routers composes the VNEXT virtual network creation and migration procedures. In order to evaluate the process, we measure the time needed to instantiate a single virtual router. Figure 4(b) shows that VNEXT creates virtual routers within 8 seconds. We consider that a virtual router is completely instantiated when the virtual router answers to ping messages. We also conduct similar tests to measure the response time for shutting down physical and virtual routers, and also to turn on a physical router using VNEXT. We observe that VNEXT takes less than 29 and 18 seconds to shutdown physical and virtual routers, respectively, and also it takes less than 39 seconds to turn on a physical router. The physical routers have more procedures to execute on kernel initialization, thus it takes more time to initialize than virtual routers. These results can be used to estimate the execution time for future policies to be used on autonomous energy saving algorithms to be implemented in VNEXT.

In the following experiment we evaluate the scalability of VNEXT, measuring the time required to collect statistics according to the number of existent virtual routers. We measure the response time of the `measureGather` controller service, that collect statistics about the physical or virtual resources, such as CPU and bandwidth usage. This service collects two samples of the infrastructure statistics for each request before generating a response. These two samples are collected in an interval of 1 second. Consequently, the response time of the `measureGather` service is always higher than 1 second. We also turn on the Graphical User Interface (GUI) to perform our experiment in a real operating scenario. The GUI periodically collects information about the network, thus generating load on the physical or virtual routers.

In our experiment we first use the `measureGatherer` service to collect statistics from the Xen Server 3. This service returns global statistics of the physical router and also statistics from the virtual routers instantiated on this server. In this case, the virtual routers statistics are collected from the physical router without directly accessing each virtual router. Figure 4(c) shows the time needed to collect statistics from the physical router according to the number of virtual routers instantiated on it. The greater the number of virtual routers, the greater the volume of collected information, thus this operation takes more time. We also evaluate the time needed



(a) Virtual network creation. (b) Router turning on and off procedure. (c) Router statistics collection.

Figure 4. Response time of virtual network creation, router turn on/off, router statistic collection VNEXT functionalities.

to collect information from a single virtual router according to the number of virtual routers instantiated in the same physical router. In this case, the virtual router is accessed to collect its information. Figure 4(c) shows that the time does not increase with the number of virtual routers because we are collecting the same statistics from a single virtual router. Furthermore, results show that the traffic generated by the operations of the GUI does not impact the physical router resources needed to perform the communication between the physical router daemons and the virtual router daemons. Finally, our results show that the system is scalable enough to take less than 4 seconds to respond to physical router measurement requests, when 128 virtual routers are instantiated. It is also worth mentioning that 1 of these 4 seconds is the time needed to obtain 2 measurement samples.

V. Conclusion

This paper presents the VNEXT system, designed to assist network administrators to make decisions. VNEXT provides specific functionalities to control and manage virtual networks, such as topology definition and monitoring, and also creation and migration of virtual routers. In addition, VNEXT offers a friendly graphical user interface, in which network managers can perform high level tasks within a few mouse clicks. We conduct experimental tests to evaluate the VNEXT system performance. Our results show that VNEXT obtains measurement responses from physical routers running 128 virtual routers in less than 4 seconds, without compromising the execution of the monitoring service. As future work, we plan to add new functionalities to VNEXT such as mechanisms to improve virtual network isolation, to provide service differentiation, to autonomously generate network profiles, and to detect violations and attacks.

Acknowledgments

The authors thank Igor M. Moraes, Marcelo D. D. Moreira, Lyno Henrique G. Ferraz, Rafael S. Alves, Tiago N. Ferreira, Victor P. da Costa, Carlo Fragni, Luciano V. dos Santos, and Renan A. Lage for their contribution.

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the ACM SOSP19*, 2003, pp. 164–177.

[2] K. Begnum, "Managing large networks of virtual machines," in *Proceedings of the 20th Large Installation System Administration Conference (LISA'06)*, 2006, pp. 205–214.

[3] A. Berl, N. Race, J. Ishmael, and H. de Meer, "Network virtualization in energy-efficient office environments," *Computer Networks*, vol. 54, no. 16, pp. 2856–2868, 2010.

[4] 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.

[5] O. Braham, A. Amamou, and G. Pujolle, "Virtual network urbanization," *Communications: Wireless in Developing Countries and Networks of the Future*, pp. 182–193, 2011.

[6] R. S. Couto, M. E. M. Campista, and L. H. M. K. Costa, "XTC: a throughput control mechanism for Xen-based virtualized software routers," in *Proceedings of the IEEE GLOBECOM'11 - accepted for publication*, Houston, Texas, USA, Dec. 2011.

[7] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, L. Mathy, and P. Papadimitriou, "Forwarding path architectures for multicore software routers," in *Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO'10)*, 2010, p. 3.

[8] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley, "Evaluating Xen for router virtualization," in *Proceedings of 16th IEEE International Conference on Computer Communications and Networks (ICCCN'07)*, 2007, pp. 1256–1261.

[9] I. Fajjari, N. A. Saadi, G. Pujolle, and H. Zimmermann, "VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic," in *Proceedings of IEEE International Conference on Communications (ICC'11)*, Jun. 2011.

[10] 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.

[11] 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, no. 5-6, pp. 339–355, Jun. 2011.

[12] J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, and M. Chiang, "DaVinci: dynamically adaptive virtual networks for a customized Internet," in *Proceedings of the ACM CoNEXT 2008*, Dec. 2008.

[13] LxCenter, "HyperVM - OpenVz, Xen, Windows Virtualization Manager," Available in http://lxcenter.org/software/hypervm. Accessed in July, 2011, 2011.

[14] N. Niebert, S. Baucke, I. El-Khayat, M. Johnsson, B. Ohlman, H. Abramowicz, K. Wuenstel, H. Woesner, J. Quittek, and L. Correia, "The way 4ward to the creation of a future internet," in *Proceedings of the 19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'08)*, 2008, pp. 1–5.

[15] P. Pisa, N. Fernandes, H. Carvalho, M. Moreira, M. Campista, L. Costa, and O. Duarte, "Openflow and Xen-based virtual network migration," *Communications: Wireless in Developing Countries and Networks of the Future*, pp. 170–181, 2011.

[16] P. Szegedi, S. Figuerola, M. Campanella, V. Maglaris, and C. Cervelló-Pastor, "With evolution for revolution: managing FEDERICA for future internet research," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 34–39, 2009.

[17] S. Vaughan-Nichols, "New approach to virtualization is a lightweight," *Computer*, vol. 39, no. 11, pp. 12–14, 2006.

[18] Z. Wang, Y. Chen, D. Gmach, S. Singhal, B. Watson, W. Rivera, X. Zhu, and C. Hyser, "Appraise: Application-level performance management in virtualized server environments," *IEEE Transactions on Network and Service Management*, vol. 6, no. 4, pp. 240–254, 2009.