Multistage Software Routers in a Virtual Environment

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Multistage Software Routers in a Virtual Environment

Andrea Bianco, Robert Birke, Luca Giraudo and Nanfang Li
Dipartimento di Elettronica, Politecnico di Torino, Italy
Email: {andrea.bianco,robert.birke,luca.giraudo,nanfang.li}@polito.it

***Abstract*—Open source routers (OSR), i.e. routers running on commodity personal computers (PC), represent a valid alternative to proprietary hardware routers. However, they may suffer from performance impairments and software limitations. Multistage architectures, based on the interconnection of elements running on standard PCs, improve single stage OSR performance. Virtualization technologies may permit to make a further step towards performance improvement (by aggregation of multiple elements into a single logical unit), increased flexibility (e.g. scalability, maintenance, consolidation) and easier introduction of new features (e.g. energy saving mechanisms). In this paper we study a previously proposed multistage architecture and consider its implementation when using virtual machines as internal components. In our experiments we demonstrate the feasibility of the architecture, and discuss some issues related to performance and architecture control.**

I. Introduction

OSRs represent an appealing alternative to proprietary network devices because of the wide availability of multi-vendor PC hardware, their low cost, the continuous performance evolution driven by the PC-market economy of scale and the large availability of open-source software for networking applications, such as Linux, BSD, Click Modular Router, XORP and Quagga.

Indeed, despite the limitations of bus bandwidth, CPU and memory-access speed, current PC-based routers have a traffic-switching capability in the range of some Gigabits per second, which is more than enough for a large number of applications. Furthermore, keeping this in perspective, performance limitations are compensated by the natural PC architecture evolution, driven by Moore's law.

However, high-end performance cannot be easily obtained today with PC-based routers: significant research efforts are on-going either to optimize the internal architecture of OSR or to devise strategies to aggregate software routers to build more powerful routing units [1]–[7].

To overcome some of the limitations of OSRs based on a single PC, A. Bianco et al. proposed to create a large size OSR exploiting a multistage switching architecture [8]–[11] to overcome issues such as unsatisfactory forwarding performance and limited number of ports. Performance measurements show that routing capabilities may scale up almost linearly with the number of internal elements. Furthermore, implementation of recovery mechanisms into the management plane can increase router resilience to close the gap with carrier-grade routers.

The multistage router defined in [8] is organized in three stages, characterized by three different internal elements: first stage load balancers (LB), interconnecting switches, and third stage back-end routers (R). In the first stage, LBs permit to scale the number of interfaces, mask the internal router structure to external routers and hosts, and balance the incoming traffic load by sending incoming packets to selected back-end routers according to a round-robin- or hash-based balancing scheme. The second stage is composed by one or more Ethernet switches that implement a logical full-mesh among elements in the first and third stage. Finally, the third stage is composed by back-end routers that forward packets at the IP layer. An internal control protocol named DIST [9] runs to manage the architecture, to identify internal elements, to configure LBs and Rs, to distribute and synchronize routing tables among Rs and to introduce features such as energy saving mechanisms based on router load monitoring.

Virtualization techniques may become an asset in networking technologies in general and in the field of distributed router architectures in particular, as recognized by several researchers [12]–[14]. Instead of buying high-end proprietary hardware-based routers, an ISP or a network administrator could either manage or even rent logical elements running on VMs (Virtual Machines) to build either a distributed router architecture. i.e., a centrally controlled interconnection network, or a more classical meshed network of routers. This enables on the one hand to re-use and share the existing computing power and on the other hand to flexibly size the router capacity to adapt it to traffic needs, enabling the introduction of energy-aware control techniques.

Three main advantages of virtualization can be highlighted:

- **larger scalability**: new internal elements can be deployed in a seamless way when traffic increases or more interfaces are needed. This enables renting of resources from data center servers, for example when new VMs are needed to add forwarding capacity.
- **easier management and reliability**: migration of VMs during maintenance periods can be implemented and faster reaction to failures should be expected by booting new VMs on general purpose servers.
- **slicing**: sharing of the same physical infrastructure among different multistage routers possibly dedicated to different types of traffic (e.g., logical separation of the operational and of the experimental networks).

As an example, we report in Fig. 1 a use case referring to an enterprise router based on the multistage architecture

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

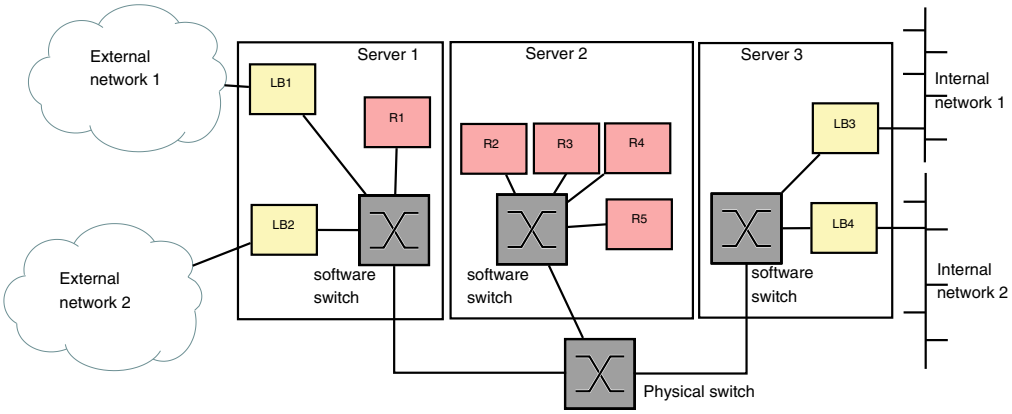

Fig. 1. Virtualization use case in multistage router architecture: three physical servers hosts different virtual machines used to build one enterprise router.

under study. External and internal network connections are terminated into the computing server farm, where virtual machines act as LBs, switches or routers. This solution permits i) to locate virtual machines on different physical servers to upgrade the overall routing capacity; ii) to share the same physical server for several virtual machines to increase resource utilization; iii) to build the multistage architecture in a mixed approach exploiting both virtual and physical elements; iv) to deploy consolidation mechanisms, e.g. move all routers to a physical server and turn off unused servers during low traffic periods.

The above described virtualization features improve performance and flexibility of the multistage architecture. However, some issues need more investigation:

- **performance penalties** due to hardware abstraction and resource contention which may impact network performance too [15];
- **additional complexity** for the management plane, due to the joint presence of physical and virtual resources;
- **larger latency** due to the introduction of additional virtualization layers.

In this paper we focus the attention on assessing the feasibility of building the multistage architecture exploiting VMs and on the identification of performance impairments due to the use of virtualization techniques. The remainder of the paper is organized as follows. In Sec. II we describe the actual implementation of the multistage architecture in some popular virtualization frameworks. Then, in Sec. III we report experimental results first on the internal elements implemented in VMs and later on the full multistage architecture. Finally, we conclude the paper in Sec. IV.

II. Multistage Router Architecture Implementation

We wish to test the feasibility of building the multistage architecture in a virtual environment, where all internal elements are running on VMs. In the original implementation, FPGA-based LBs were also considered to improve balancing speed. However, in this paper we limit our analysis to software LB implementations based on Click to exploit VMs. The central

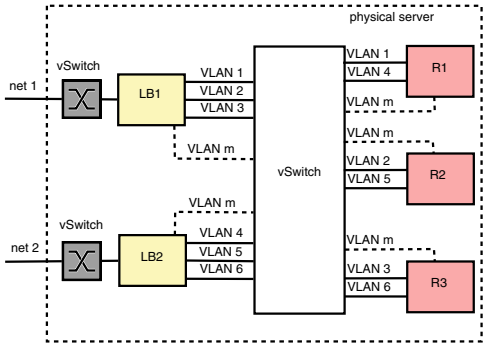

Fig. 2. Internal configuration of a VMware based multistage router architecture using a per-port VLAN tagging.

stage of the router is composed by an Ethernet switch which has to support multicast and unicast traffic. The software switch usually provided by virtualization platforms can be adopted. Finally, routers are VMs running Linux OS (Operating System) and XORP, without any particular functional limitations or implementation constraints.

We considered two major virtualization frameworks: XEN and VMware ESXi version. Both tools provide similar functionalities (e.g. full virtualization or para-virtualization depending on CPU features) using different approaches. XEN is an open-source project based on the Linux kernel, meanwhile VMware is a closed-source project. Since VMware provides better performance [16], we focus on the VMware implementation of the three key elements, i.e., LBs, switch and Rs, on a single physical high-end PC running the virtualization software.

VMware is based on a closed source approach; thus, it is difficult to obtain details about internal networking functionalities and available features. We rely on the *vSphere* management interface and VMware drivers.

VMware ESXi is targeted at server virtualization which is the free version of the VMware server program suite. Thus, it shows some limitations on internal VMs management due to design constraints and security reasons (e.g. to guarantee VM isolation). As a result, the implementation of the multistage

router in this scenario is more difficult, since the internal software switch (named *vSwitch*) is not behaving as a standard Ethernet switch. Indeed, vSwitch does not implement the spanning tree algorithm and the backward address learning mechanism.

The vSwitch has two operational modes:

1) **promiscuous mode on (hub)**: packet broadcasting.
2) **promiscuous mode off (switch)**: at most one MAC address is associated with a switch port. Thus, packets are forwarded to this destination only.

From the implementation point of view, the first operational mode is not appropriate, mainly because of performance degradation due to an excessive use of packet broadcasting. The second is the most appropriate, but the limited vSwitch functionalities do not permit to support LBs behaviour. Indeed, LBs receive from Rs packets not addressed to their internal MAC address but to the MAC address of external devices. However, all packets with a MAC destination address different from the LB's MAC address are discarded by the vSwitch (in *switch* mode).

According to our understanding it is not possible to configure the vSwitch to behave as a standard Ethernet switch. Thus, we defined a workaround to make the system work in a proper way. Two possible solutions can be envisioned:

- **hub config**: use as many two-port hubs (promiscuous mode on) as the number of interconnections among LBs and Rs, to create point-to-point links between them. Every hub connects one LB and one R only.
- **VLAN config**: use one vSwitch, as in Fig. 2, and configure one VLAN for each R-LB pair.

Both solutions are equivalent from the functional point of view and permit to implement a fully-functional multistage router. In both cases an additional hub (or switch) is needed to interconnect all LBs and Rs with a full-mesh network to permit normal operations of the DIST control plane (e.g. VLAN m in Fig. 2).

III. Experimental setup and results

All experiments were run on Dell Power Edge T100 servers equipped with an Intel Xeon E3110 running at 3.0 GHz with `pae` and `vmx` flags enabled, 2 cores, 8 GB DDR2 RAM and 2 Intel pro1000 PCIe network interface cards (NIC). The chosen hypervisor is VMware ESXi 4.0. VMs run Ubuntu 9.04 with Linux kernel 2.6.28-11-generic. Traffic is generated and received by an Agilent N2X router tester (Gigabit Ethernet module) which is connected to the server on two different interfaces. Graphs report the average of five independent runs; the performance difference among the five tests is negligible. When single elements are tested, we use one vSwitch to directly connect VMs to one physical NIC; in the case of the multistage router, a more complex configuration scheme is needed, as previously described.

Every graph reports routing performance of the physical machine running the Linux OS (1 or 2 active cores, identified by the *Phy* prefix in the plots) to provide an upper bound

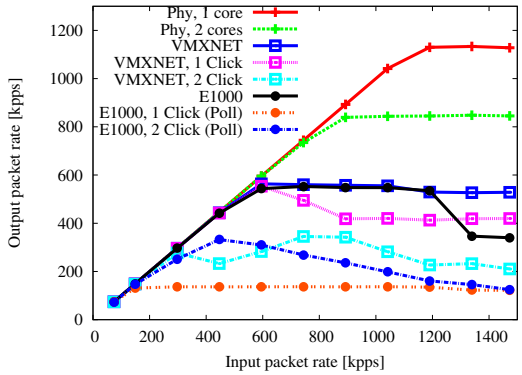

Fig. 3. Performance evaluation of a Click-based LB into VMware ESXi 4.0 environment using 64 bytes packets.

to VM forwarding performance. After preliminary test, we assessed that VMware, contrary to Linux, performs better with both CPUs active. Thus, all tests with VMs use two CPUs, unless otherwise stated.

A. Load balancers performance

VMware ESXi exports to the guest OS different virtual network interfaces. We consider here only VMXNET, giving the best performance, and the Intel e1000 due to the availability of performance enhancing Click patches. The VMXNET is a custom VMware network interface. Thus, additional drivers are needed to use it. In the second case, a virtual Intel e1000 hardware is emulated and the standard Linux driver is used (in some cases with the addition of the e1000-related Click patch).

We report in Fig. 3 a comparison among different virtual network drivers when running LBs in VMs. One physical interface is used to connect the PC to the router tester, whereas VMs are connected to a single vSwitch directly connected to the physical interface.

Regardless of the chosen NIC driver, virtualization introduces a large overhead. In the best case of the VMXNET driver, throughput drops to about 60% of reference throughput of physical servers. Furthermore, using Click and/or sharing resources among LBs introduces additional overheads. For instance, in the case of 2 VMXNET-based LBs, throughput drops to approximately 25% of the reference throughput. This is most likely due to context switching and resource contention (cache misses and interrupts management). Indeed, the cost of context switching is higher when more VMs are sharing the same resources, because more VMs contend for the same resources and the VM state has to be restored each time before execution starts. Thus, sharing physical resources among different virtual LBs significantly limits performance.

Finally, the e1000 virtual NIC obtains worse performance than those of VMXNET optimized drivers. Even when using the polling patch (NAPI-aware) in Click Load-balancers, throughput is still unsatisfactory. Since NAPI eliminates the negative effect of interrupt trashing, performance decrease should be ascribed to VMs resource contention and VM state

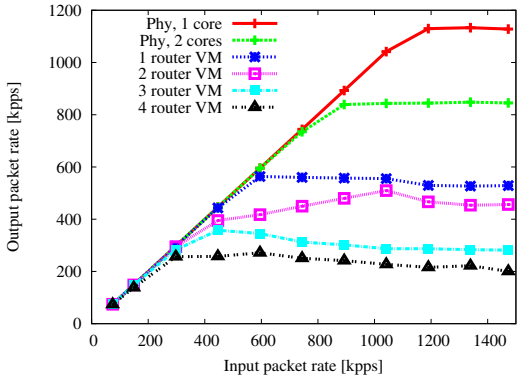

Fig. 4. Performance evaluation of routers in VMware ESXi 4.0, using VMware's VMXNET driver and 64 bytes packets.

TABLE I
FAIRNESS TESTS: THROUGHPUT OF TWO 64-BYTE-PACKETS FLOWS WHEN VARYING THE FLOW RELATIVE LOAD.

Total input load: 50%				Total input load: 100%			
flow 1		flow 2		flow 1		flow 2	
in	out	in	out	in	out	in	out
0.5	68.3%	0.5	64.4%	0.5	37.6%	0.5	39.2%
0.3	42.1%	0.7	97.5%	0.3	17.9%	0.7	47.9%
0.1	75.0%	0.9	97.5%	0.1	13.3%	0.9	42.9%

suspend/restore activities, which have a severe negative impact on the routing performance, especially at high loads.

B. Back-end routers performance

We run experiments using the same internal configuration described in the previous section, but running routers instead of Click-based LBs. Only VMXNET NICs are considered due to their better performance. Results are reported in Fig. 4: also in this case the virtualization overhead is significant (roughly 30% in performance reduction). Resource sharing among back-end routers has also a negative impact on aggregated throughput, as shown when increasing the number of active VMs concurrently running on the same PC. The performance drop is smaller than in the previous case. Indeed, the aggregated throughput drops to 25% when four routers are used at the same time. Results not reported show that the impact of resource sharing depends on the VM activity level: indeed, the performance of a single active VM does not change when increasing the number of concurrently running VMs from one to three, if the other VMs are idle, i.e., they do not route packets.

Beside throughput measurements, we consider fairness issues too. We generate 2 flows towards 2 back-end Rs using different load distributions. Results are reported in Tab. I. The in column is the input load share of the flow, while the out column shows the percentage of packets forwarded for the considered flow. In overload, VMware ESXi does not isolate well network flows, the smaller flow being significantly penalized.

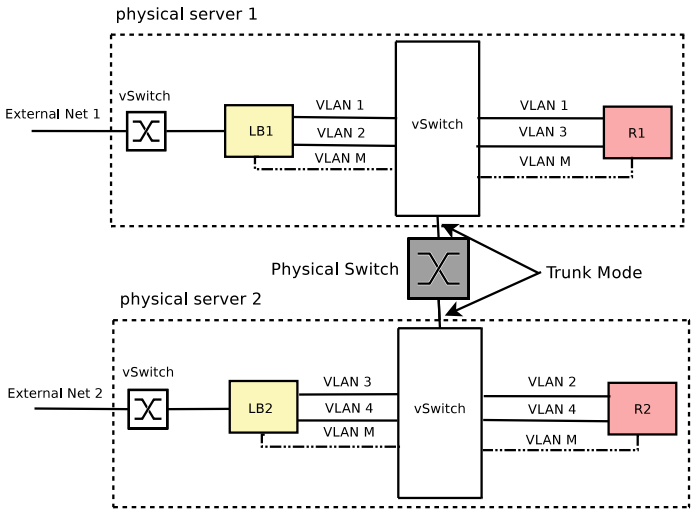

Fig. 5. The implementation of the Multistage software routers in the 2 physical server scenario.

C. Multistage router performance

We test a full multistage router architecture considering three simple cases:

1) one physical server, one LB and one R (labelled 1L+1R)
2) one physical server, two LBs and two Rs (labelled VLAN-based, hub-based and static ARP)
3) two physical servers, one LB and one R per server (labelled 2 servers)

In the first case, which represents the minimal configuration, we consider the hub configuration only for the internal vSwitch, since more complex configurations are not needed, whereas in the second case we compare three different internal configurations: the VLAN-, the hub- based solutions, as described in Sec. II, and a *static ARP* solution consisting in a simple ARP table modification needed in back-end routers that permits to use vSwitch in *switch* mode. This last configuration, used only as a reference, provides correct performance indications, but implementation-wise would create wrong packet addressing at the MAC level. Finally, in the third case we consider performance scale with additionally deployed resources. Only in this case we use the VLAN-based solution described in Fig. 5.

Performance results are reported for 64, 512 and 1500 byte packets in Fig. 6. Throughput is rather poor, especially for small packet size. This is expected, due to the already observed performance limitations of single elements induced by virtualization and resource sharing (e.g. frequent interruptions to run different VMs, context switching and execution status restoration). Indeed, in the first case (one LB and one R) we obtain better performance than in the second case, even reaching wire speed for large packet size.

In the second case, regardless of the internal configuration, since more elements (two LBs and two Rs) share the same resources, the CPU becomes the bottleneck, and the upper

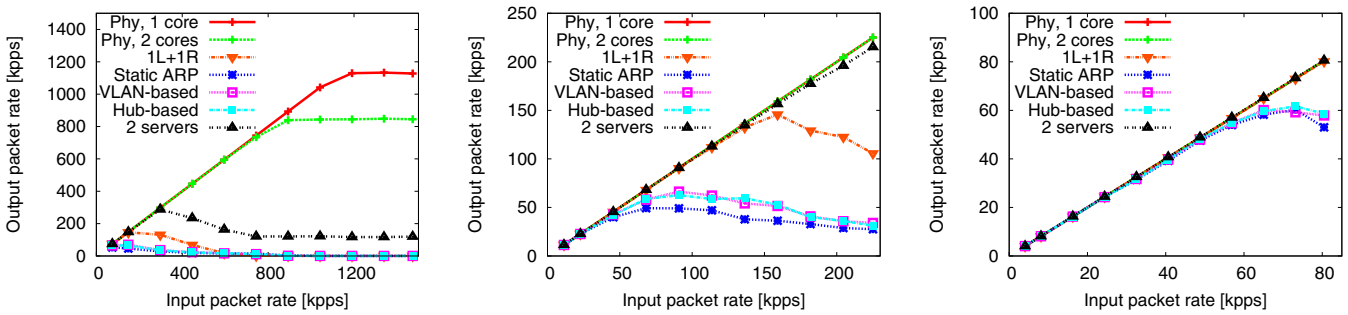

Fig. 6. Performance evaluation of multistage router (2 LB + 2 OSR) into VMware ESXi 4.0 environment with different internal configurations: from left to right, 64, 512 and 1500 bytes packets.

performance limit is around 70 kpps, which is not enough to reach wire speed even for large packet size.

Running the multistage in two physical servers leads to performance improvements, which are still unsatisfactory for 64 bytes packets, but that permits to reach wire-speed starting from a packet size of roughly 512 bytes. This is due to the larger amount of deployed resources and to reduced contention, as in the first case.

Finally, no major performance differences can be observed among the various internal networking configurations: thus, the utilization of hubs or VLAN tagging functionalities does not influence performance in the studied scenario, where the bottleneck is CPU overload.

IV. Conclusions and Future Work

We demonstrate the feasibility of deploying a multistage router architecture in a virtualized environment. We highlighted potential advantages of this solution, mainly in terms of flexibility and scalability. The correctness of the overall architecture has been verified via a practical implementation on VMs of the internal elements needed to build the multistage router. We highlighted strong performance limitations that makes today this approach rather difficult to pursue with the current level of virtualization technology. To improve performance, research work is needed in many areas, e.g.:

- **hypervisors**: reduction of overhead costs and optimization of NIC virtualization, solving isolation/fairness issues (mainly on CPU and NICs sharing).
- **multistage control plane**: optimization of VM allocation on different physical servers, to reduce as much as possible resource sharing.
- **virtualization approaches**: selection of less invasive virtualization approaches like OS level virtualization (e.g. OpenVZ and Linux Vserver) and minimized OS images for virtual machines; an interesting approach to be considered is Denali OS [17], where hypervisor and OS are designed from scratch to tightly collaborate to reduce virtualization overhead.

Nevertheless, the natural evolution of PCs in many areas (e.g. CPU speed, multi-queue support in NICs, networking stack) makes us confident that this approach will obtain acceptable performance in the next future (for example see [7]), increasing the interest of the scientific community in this research area.

Acknowledgments

These activities were developed in the framework of the FEDERICA project, funded by the European Commission.

References

[1] R. Bolla and R. Bruschi, "RFC 2544 performance evaluation and internal measurements for a Linux based open router," in *HPSR*, Poznan, Poland, Jun. 2006.
[2] ——, "PC-based software routers: high performance and application service support," in *PRESTO*, Seattle, USA, Aug. 2008.
[3] ——, "An effective forwarding architecture for SMP Linux routers," in *IT-NEWS*, Venice, Italy, Feb. 2008.
[4] K. Argyraki, S. Baset, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, E. Kohler, M. Manesh, S. Nedevschi, and S. Ratnasamy, "Can software routers scale?" in *PRESTO*, Seattle, USA, Aug. 2008.
[5] IETF, "Forwarding and control element separation working group (ForCES)," http://tools.ietf.org/wg/forces/.
[6] W. Wang, L. Dong, B. Zhuge, M. Gao, F. Jia, R. Jin, J. Yu, and X. Wu, "Design and implementation of an open programmable router compliant to IETF ForCES specifications," in *ICN*, Sainte-Luce, Martinique, Apr 2007.
[7] M. Dobrescu, N. Egi, K. Argyraki, B. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: exploiting parallelism to scale software routers," in *SOSP*, Big Sky, USA, Oct. 2009.
[8] A. Bianco, J. Finochietto, M. Mellia, F. Neri, and G. Galante, "Multistage switching architectures for software routers," *IEEE Network*, vol. 21, no. 4, pp. 15–21, Jul.-Aug. 2007.
[9] A. Bianco, R. Birke, J. M. Finochietto, L. Giraudo, F. Marenco, M. Mellia, A. Khan, and D. Manjunath, "Control and management plane in a multi-stage software router architecture," in *HPSR*, Shanghai, China, May 2008.
[10] A. Khan, R. Birke, D. Manjunath, A. Sahoo, and A. Bianco, "Distributed PC based routers: bottleneck analysis and architecture proposal," in *HPSR*, Shanghai, China, May 2008.
[11] A. Bianco, J. Finochietto, G. Galante, M. Mellia, D. Mazzucchi, and F. Neri, "Scalable layer-2/layer-3 multistage switching architectures for software routers," in *IEEE GLOBECOM*, San Francisco, USA, Dec. 2006.
[12] "FEDERICA project," http://www.fp7-federica.eu/.
[13] M. B. Anwer and N. Feamster, "Building a fast, virtualized data plane with programmable hardware," in *VISA*, Barcelona, Spain, Aug. 2009.
[14] M. Caesar and J. Rexford, "Building bug-tolerant routers with virtualization," in *PRESTO*, Seattle, USA, Aug. 2008, pp. 51–56.
[15] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley, "Evaluating Xen for router virtualization," in *ICCCN 2007*, Honolulu, USA, Aug. 2007.
[16] N. Li, "Multistage Software Routers Implementation in Virtual Environment," Master's thesis, Politecnico di Torino, Torino, Italy, 2009.
[17] A. Whitaker, M. Shaw, and S. D. Gribble, "Scale and performance in the Denali isolation kernel," *SIGOPS*, vol. 36, pp. 195–209, Dec. 2002.