POLITECNICO DI TORINO
Repository ISTITUZIONALE

Scalable Layer-2/Layer-3 Multistage Switching Architectures for Software Routers

Original
Scalable Layer-2/Layer-3 Multistage Switching Architectures for Software Routers / Bianco, A., Finochietto, J., Galante, G., Mazzucchi, D., Mellia, M., Neri, F.. - STAMPA. - (2006). (IEEE GLOBECOM 2006 San Francisco, CA, USA 27-30 November 2006) [10.1109/GLOCOM.2006.343].

Availability:
This version is available at: 11583/1532085 since:

Publisher:
IEEE

Published
DOI:10.1109/GLOCOM.2006.343

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

19 November 2026

Scalable Layer-2/Layer-3 Multistage Switching Architectures for Software Routers

Andrea Bianco*, Jorge M. Finochietto*, Giulio Galante†,
Marco Mellia*, Davide Mazzucchi*, Fabio Neri*
* Dipartimento di Elettronica, Politecnico di Torino, 10129 Torino, Italy, Email: {firstname.lastname}@polito.it
† Networking Lab, Istituto Superiore Mario Boella, 10138 Torino, Italy, Email: galante@ismb.it

***Abstract*— Software routers are becoming an important alternative to proprietary and expensive network devices, because they exploit the economy of scale of the PC market and open-source software. When considering maximum performance in terms of throughput, PC-based routers suffer from limitations stemming from the single PC architecture, e.g., limited bus bandwidth, and high memory access latency. To overcome these limitations, in this paper we present a multistage architecture that combines a layer-2 load-balancer front-end and a layer-3 routing back-end, interconnected by standard Ethernet switches. Both the front-end and the back-end are implemented using standard PCs and open-source software. After describing the architecture, evaluation is performed on a lab test-bed, to show its scalability. While the proposed solution allows to increase performance of PC-based routers, it also allows to distribute packet manipulation functionalities, and to automatically recover from component failures.**

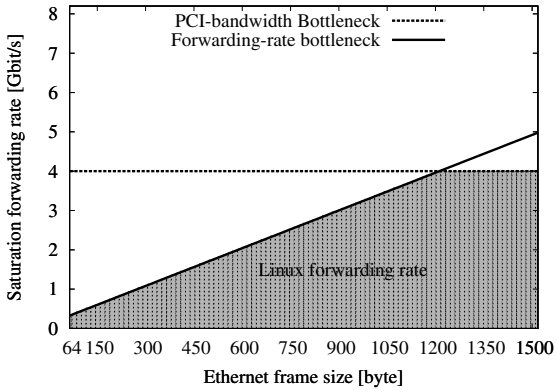


Fig. 1. Saturation forwarding rate vs. Ethernet-frame size for a Linux router using commercial NICs

I. Introduction

Software routers based on off-the-shelf personal-computer (PC) hardware and open-source software are becoming appealing alternatives to proprietary network devices because of the wide availability of multi-vendor hardware, the low cost and the continuous evolution driven by the PC-market economy of scale. Indeed, the PC world benefits from the de-facto standards defined for hardware components, which enable the development of an open multi-vendor market, and the large availability of open-source software for networking applications, such as Linux [1], Click [2] and the BSD derivatives [3] for the data plane, as well as Xorp [4] and Zebra/Quagga [5] for the control plane.

Several criticisms can be raised against software routers, e.g., software limitation, lack of hardware support, scalability problems, lack of advanced functionalities; even though, performance limitations are compensated by the natural PC architecture evolution. Current PC-based routers and switches have a traffic-switching capability in the range of a few gigabits per second, which is more than enough for a large number of applications. However, when looking for high-end performance, PC-based routers are affected by many limitations. In [6], where commercial Network Interface Cards (NIC) were used to build a router running both the standard Linux and the Click Internet Protocol (IP) stack, we showed that it is not possible to route a single 1-Gbit/s traffic flow consisting of only minimum-size Ethernet[1] frames, even if the PCI bus bandwidth is 8 or 16 Gbit/s. The main limitations stem from central-processing unit (CPU) overloading and from large memory-read latency. In this paper we consider Linux kernel as the software router Operating System.

Indeed, the performance of PC-based routers is mainly limited by two factors, as shown in Fig. 1, which models the saturation forwarding rate for a PC with a 8-Gbit/s PCI bus versus the Ethernet-frame size. First, for small-size packets, the *forwarding-rate bottleneck*, stemming from the maximum packet rate (which was observed to be 640 kilo-packets per second (kpps) in [6]) that the architecture can forward because of CPU availability and memory-read-latency constraints. Second, for large-size packets, the PCI-bus maximum bandwidth. The 4-Gbit/s figure is due to the internal PC architecture, which forces packets to go twice through the 8-Gbit/s PCI bus. Therefore, while a rough capacity of 8-Gbit/s is available, the real capacity of a high-end PC is limited to the operating area highlighted by the shadowed pattern in Fig. 1.

Moreover, another important limitation of current PC based architectures is due to the number of interfaces that can be hosted in a single PC. Indeed, the number of PCI slots is usually limited to 5 or 6 slots, and the number of Ethernet ports per PCI card is limited to 4, therefore bounding to about 24 the number of ports a single PC router can host. Moreover,

[1] In this paper we refer to both "Ethernet 2.0" and "IEEE802.3" as "Ethernet".

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.
Authorized licensed use limited to: Politecnico di Torino. Downloaded on April 27,2010 at 07:24:26 UTC from IEEE Xplore. Restrictions apply.

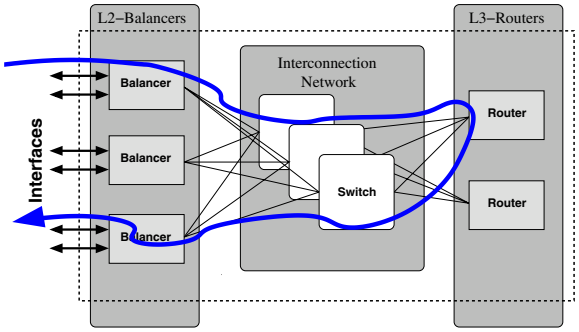


Fig. 2. Schematic of the multistage architectures

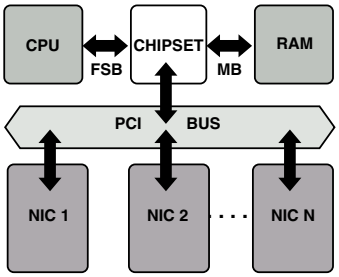


Fig. 3. Key components in a PC-based software router

by sharing a single PCI slot, multi-port PCI cards usually face additional performance limitations.

Multistage architectures [7] have been studied to overcome the single machine limitations. Initially studied in the context of circuit oriented networks, they have been traditionally used in the design of parallel computer systems, and lately considered a viable mean to build large switching elements [8], [9].

To overcome the single PC limitations, in this paper we propose a multistage bidirectional architecture, in which layer-2 (Ethernet) switching and layer-3 (IP) routing capabilities are distributed among stages. Fig. 2 sketches the proposed architecture. The idea is to use a front-end stage made of a set of PCs that act as simple layer-2 switching elements that load balance incoming packets. A back-end stage comprising a second set of PCs is then used to perform more complex operations, e.g., layer-3 forwarding, classification, filtering, accounting, etc. We will refer to the first-stage elements as *L2-balancers*, while the back-end elements will be referred to as *L3-routers*. The two stages are interconnected by means of one (or more) layer-2 switch, which offers a logically fully-connected mesh between each balancer and each router.

The goal of the balancers is to uniformly distribute packets to the back-end routers by correctly addressing (at layer-2) the back-end NIC addresses. Considering the back-end routers, they act as traditional layer-3 routers, which receive, manage and forward packets using the traditional IP paradigm.

The proposed architecture allows to overcome the performance limit of a single router by offering parallel paths. It also allows to scale the total NIC number the node can host, by offering a total number of PCI slots which grows linearly as the number of L2-balancers. In addition, the multistage architecture can be exploited to automatically recover from faults, i.e., reconfiguration can occur in case of any PC/switch failure.

In this paper we evaluate the scalability of this architecture by measurements using a laboratory test-bed. Functionalities distribution across layer-3 routers could be obtained as an additional advantage of the proposed architecture, but it is beyond the scope of this paper. Similarly, issues related to the management of the architecture, i.e., to the "control plane" of an IP router, are not discussed.

The rest of the paper is organized as follows. Sec. II gives a quick introduction to the PC architecture, describes the operations and the bandwidth limitations of its key components, and explains how a PC can be used as an IP router and summarizes its performance limits. Sec. III describes the main features of the multistage architecture we propose. Sec. IV introduces the experimental setup, describes the performed tests, and comments on the obtained results. Finally, Sec. V concludes the paper.

II. Software Router Architecture

A. PC Hardware architecture

A PC comprises three main building blocks: central processing unit (CPU), random access memory (RAM), and peripherals, glued together by the *chipset*, which provides complex interconnection and control functions.

As sketched in Fig. 3, the CPU communicates with the chipset through the front-side bus (FSB). The RAM provides temporary data storage for the CPU, and can be accessed by the memory controller integrated on the chipset through the memory bus (MB). The NICs are connected to the chipset by the PCI shared bus.

Today's state-of-the-art CPUs run at frequencies up to 3.8 GHz. High-end PCs are equipped with chipsets supporting multiple CPUs connected in a symmetric multiprocessing (SMP) architecture. Typical configurations comprise 2, 4, 8 or even 16 identical CPUs.

The front-side bus is 64-bit wide and is driven by a 100- to 266-MHz *quad-pumped* clock, allowing for a peak transfer rate ranging from 3.2 Gbyte/s to 8.4 Gbyte/s.

The memory bus is usually 64-bit wide and runs at 100, 133, 166, or 200 MHz with *double-pumped* transfers, providing a peak transfer rate of 1.6, 2.1, 2.7, or 3.2 Gbyte/s. In high-end PCs, the memory bandwidth is further doubled, bringing the bus width to 128 bits, by installing memory banks in pairs. Note that this allows to match the memory-bus peak bandwidth to that of the front-side bus.

The PCI protocol is designed to efficiently transfer the contents of large blocks of contiguous memory locations between the peripherals and the RAM, without requiring any CPU intervention, i.e., using Direct Memory Access (DMA). Depending on the PCI protocol version implemented on the chipset and the number of electrical paths connecting the components, the bandwidth available on the bus ranges from

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.
Authorized licensed use limited to: Politecnico di Torino. Downloaded on April 27,2010 at 07:24:26 UTC from IEEE Xplore. Restrictions apply.

about 125 Mbyte/s for PCI 1.0, which operates at 33 MHz with 32-bit parallelism, to 4 Gbyte/s for PCI-X 266, when transferring 64 bits on a double-pumped 133-MHz clock.

Typically, Ethernet NICs operate as bus-masters to offload the CPU from performing bulk data transfers between their internal memory and the RAM. Incoming packets are first buffered on the NIC, then transferred to to RAM in DMA through the PCI bus, and packet descriptors are stored in ring data structures implemented in RAM. Each NIC is connected to one interrupt-request (IRQ) line, that is used to notify the CPU of events that need service from the operating system.

Summarizing, common PC hardware enables to easily implement a shared-bus, shared-memory router, where NICs receive and transfer packets directly to the RAM, the CPU routes them to the proper output ring buffer in RAM, and NICs fetch packets from the RAM and transmit them on the wire. In such configuration, each packet travels twice through the PCI and the memory bus, effectively halving the bandwidth available for routing traffic. Therefore, a high-end PC equipped with a PCI-X bus should in principle be able to offer a backplane throughput of up to 16 Gbit/s.

B. Single PC Performance

In this section, we briefly summarize the performance limitation a single PC suffers when used as software router. Readers interested in a deeper performance evaluation can refer to [6]. We consider as baseline system a PC equipped with PCI-X Intel PRO/1000 linecards, a single Intel Xeon CPU running at 2.6 GHz, 1 Gigabyte double-pumped, 128-bit wide, 200 MHz DDR RAM and PCI-X bus running at 133 MHz. We consider Linux kernel version 2.6.12 as reference architecture. NAPI is enabled at the NIC driver. An Agilent N2X RouterTester 900 [10], equipped with 8 Gigabit-Ethernet ports, which can transmit and receive Ethernet frames of any size at full rate, was used to generate traffic as well as sourcing and sinking traffic in routing tests.

Fig. 4 reports the throughput in packets per second (pps) when a single flow is loading the router, i.e., packets enter the router from a single Gigabit-Ethernet NIC and have to be routed toward a different Gigabit-Ethernet NIC. Both the theoretical throughput (dashed curve), and the measured throughput (solid line) are reported, highlighting the impact of optimized buffer management (black dots) or standard memory allocation (white dots). The plot clearly shows the impact of packet size on the performance, showing that a single PC can only reach about 640 kpps considering 64-byte-long minimum-size Ethernet frames. As pointed out in [6], the limit stems from the high latency faced by the output NIC when requesting (via a DMA operation) a packet from the main memory. Moreover, when more complex operations have to be performed by the CPU, e.g., imposing Access Control List (ACL) rules, Network Address Translation (NAT) operations, etc., the per-packet processing time can increase so as to reduce even more the maximum throughput a single PC can sustain. Since the optimized memory management is shown to guarantee (slightly) better performance, in the following tests we will always adopt it.

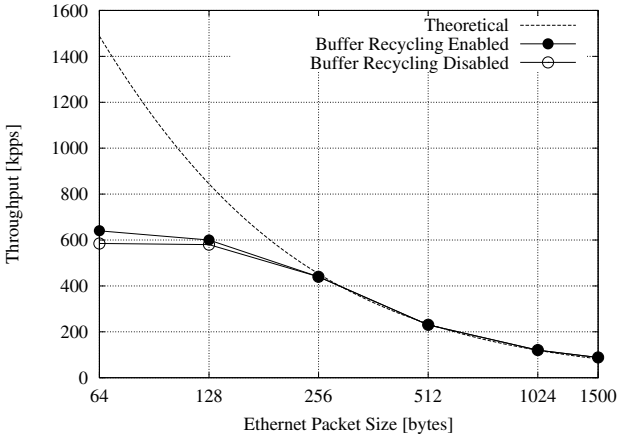


Fig. 4. Software Router Performance

III. Multistage Architecture

The previously exposed limitations of a single PC drove the design of a multistage architecture. The idea is to have a first-stage of load-balancing switches (L2-balancers), and a back-end stage of IP routers (L3-routers), which are interconnected by means of standard Ethernet switches. Both L2-balancers and L3-routers will be implemented by standard PCs equipped with several linecards, and running a (possibly modified) version of the Linux kernel. Fig. 2 sketches the architecture, in which Input/Output cards are on the leftmost part of the figure. Packets arriving at the router input ports will be

1) received by a L2-balancer front-NIC, processed by its CPU to perform simple and fast load balancing among the back-end routers, then transmitted by the L2-balancer back-NIC toward the interconnection network;
2) switched by the interconnection network to the proper L3-router NIC;
3) received by the L3-router and processed by its CPU to perform the required packet operations, then transmitted toward the interconnection network;
4) switched by the interconnection network to the proper L2-balancer back-NIC;
5) received by the L2-balancer back-NIC, processed by its CPU to switch the packet toward the proper L2-balancer front-NIC, then transmitted toward the next-hop node.

Note that, being the ports of a router bidirectional ports, packets are forced to enter and exit from and to the same port.

A. L2-Load Balancer Operation

The operations at item 1) require the load balancer to forward packets from the front-NIC to the back-NIC, possibly adapting the layer-2 framing formats. Several algorithms can be implemented, from a simple round-robin scheme to more complex algorithms that, e.g., guarantee in-sequence routing

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.
Authorized licensed use limited to: Politecnico di Torino. Downloaded on April 27,2010 at 07:24:26 UTC from IEEE Xplore. Restrictions apply.

of packets [9], or balance packets to a particular L3-router based on QoS parameters. Load balancing is obtained simply by setting the destination MAC address of the Ethernet frame, so that the proper L3-router Ethernet NIC is addressed[2].

For the sake of simplicity, in the paper we consider a simple round-robin scheme, and detail three possible implementations in the Linux kernel:

ODB — Output Driver Balancing: in this case, packets follow the standard path in the Linux kernel, where they are routed according the IP rules. When a packet is scheduled for transmission to a back-NIC via the `hard_start_xmit()` function call, the load-balancing code that changes the destination MAC address according to the round-robin scheme is executed. A static *balancing table*, that stores the L3-router NIC MAC addresses, and a simple pointer-based scheme is adopted. By managing packets at the IP layer, the routing table of the L2-balancer must be configured to allow the Linux kernel to correctly route packets to the output NIC. The major advantage of this solution is that any advanced packet manipulation feature available in the Linux kernel can be adopted (e.g., filtering, QoS scheduling, etc.) before the load-balancing. Similarly, any layer-2 format can be used, as long as it is supported by the Linux kernel. The major drawback is the unneeded complexity of operations performed in the kernel.

DTX — Direct Transmission: in this case, packets received by the front-NIC are processed directly by the front-NIC driver code, which, after changing the destination MAC address according to the load balancer algorithm, calls the `hard_start_xmit()` function addressing the back-NIC directly. By managing packet at the Ethernet layer, there is no need to correctly set-up the IP routing tables. This solution minimizes the packet processing, but does not allow to exploit any advanced packet manipulation functionality. Moreover, it works only if the front-NIC layer-2 format is Ethernet based.

DTX-n — Direct Transmission with n back-NICs: this is the same scheme as DTX, but several back-NICs are used. Therefore, the input driver performs two round-robin schedules to balance packets: i) on the n back-NICs and ii) on the L3-router MAC addresses. This solution has the same advantages of the DTX one, but it permits to obtain better performance, e.g., overcoming the maximum throughput limit a single NIC solution faces.

Considering operations involved at step 5), packets received by L2-balancer back-NIC must be switched according to the destination MAC address to the corresponding front-NIC. We implemented a solution similar to the DTX one, in which the back-NIC driver has a static *forwarding table* storing next-hop MAC addresses. When a packet is received by the back-NIC driver, a look-up in the forwarding table is performed to call the proper front-NIC `hard_start_xmit()` function, causing therefore a direct transmission of the (unmodified) frame toward the next-hop. An additional entry is used to deal with broadcast/multicast messages, so that they are correctly duplicated to all front-NICs. This is useful, e.g., to correctly broadcast ARP queries an L3-router generates. Possibly, the L2-balancer can directly reply to the ARP request, by using information in the forwarding table.

[2]In case both front- and back- NICs are Ethernet NICs, a simple re-write operation of the MAC address is required.

B. Interconnection Network Operations

Operations involved at items 2) and 4) are implemented by standard Ethernet Switches, that perform normal backward learning and switching operations. Indeed the load-balancing among L3-routers is achieved by addressing the proper L3-router input NIC MAC address. Therefore, there is no need to change the normal operation of Ethernet switches.

C. L3-Router Operations

Operations involved at item 3) are implemented by L3-routers. Since classic IP routing and packet manipulation operations are used, no changes are required compared to the standard feature set a single-box router implements. All L3-routers must be correctly configured, e.g., IP routing tables, firewalling rules, etc. must be correctly set-up.

In summary, both the interconnection architecture and the L3-router stages require only standard functionalities, while the only modification involves the L2-balancer implementation. While we propose here three possible software implementations using the Linux kernel, we point out that both the load balancing and the switching capabilities a L2-balancer requires can be easily implemented in hardware, since they are very similar to the requirements of a standard L2 switch.

IV. Performance Results

We implemented the ODB, DTX and DTX-n L2-balancers in a Linux kernel version 2.6.12. We then set-up a testbed involving PCs with the same hardware configuration as the one used as baseline reference. An unmanaged 3Com "OfficeConnect" Gigabit Switch equipped with 8 ports was used to build the interconnection network. In this section we present performance measurements obtained on the resulting test-bed, considering only minimum-size Ethernet frames.

A. L2-balancer Performance

We first tested the performance of a L2-balancer by loading a front-NIC using the router tester sources, and then directly connecting the back-NIC(s) to the router tester sink(s). To stress the system, we considered only minimum-size Ethernet frames. Fig. 5 reports the results by comparing the performance of both the ODB (dark gray) and the DTX-n (black) solutions versus the number n of back-NICs. As reference values, also the theoretical maximum of 1488 kpps is reported (light gray). The ODB solution is not surprisingly limited at about 640 kpps which has been shown to be also the limit of a single PC-router. The DTX solution (i.e., DTX-1) shows little improvement, suggesting that the bottleneck is due to the memory latency and not CPU overload. Increasing the number of back-NICs to 3 guarantees to reach 100% of throughput. This results shows that it is possible to achieve

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.
Authorized licensed use limited to: Politecnico di Torino. Downloaded on April 27,2010 at 07:24:26 UTC from IEEE Xplore. Restrictions apply.

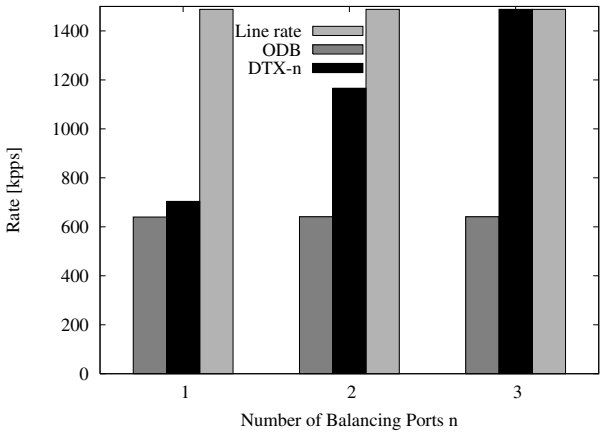


Fig. 5. Balancer Performance

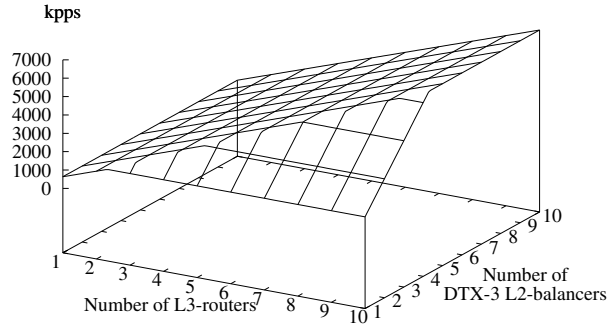


Fig. 6. Router Stage Performance

line-rate balancing when considering minimum-size frames with a software implementation, but at least 3 back-NICs are required.

B. Scaling Performance of the Multistage Router

Given that a L3-router is identical to a standard router, its performance limits have been already presented in Sec. II-B. Similarly, the switches adopted in the interconnection network proved to be able to sustain 100% of full-duplex traffic when fully loaded. We therefore can only present results of the complete multistage architecture. We set up a testbed with 2 L2-balancers and 2 L3-routers. We measured the maximum throughput performance considering minimum-size Ethernet frames, obtaining a value of 1280 kpps, which exactly matched the expected results of doubling the performance of a single PC router, i.e., 2×640 kpps. Considering minimum-size packets, Fig. 6 shows the maximum throughput (in kpps) a multistage architecture offers versus the number of L2-balancers (in DTX-3 configuration) and L3-Routers. For example, it shows that to reach a 1 Gigabit Ethernet line rate (1488 kpps), it is sufficient to use a single L2-balancer, and 3 L3-routers. Similarly, by considering a setup with 4 L2-balancers and 10 L3-routers it is possible to route 4 Gigabit Ethernet at line rate, i.e., 6 Mpps.

V. Conclusions and Future Work

Based on standard PCs and open-source software, in this paper we presented a multistage architecture that allows to overcome the performance limitation of a single software router. By combining simple layer-2 load-balancing capabilities at the front stage and an array of layer-3 routers at the back stage, the resulting architecture is shown to offer very good scalability properties, e.g., allowing to route minimum-size packets at line rate.

We are currently improving the load-balancer algorithm to allow a more general distribution of functionalities and automatic fault recovery. Similarly we are investigating how to integrate automatic control capabilities that allow to control both front- and back-stage PCs managing their configurations, e.g., automatic distribution of routing tables, packet filtering rules, etc.

Acknowledgments

This work was performed in the framework of the Bora-Bora [11] project funded by the Italian Ministry of University, Education, and Research (MIUR), and developed in the high-quality lab LIPAR at Politecnico di Torino.

References

[1] L. Torvalds, "Linux OS." [Online]. Available: http://www.linux.org
[2] E. Kohler, R. Morris, B. Chen, and J. Jannotti, "The Click modular router," *ACM Trans. on Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000.
[3] "BSD Unix." [Online]. Available: http://www.bsd.org
[4] M. Handley, O. Hodson, and E. Kohler, "Xorp: An open platform for network research," in *Proc. of the 1st Workshop on Hot Topics in Networks*, Princeton, NJ, US, Oct. 28–29, 2002.
[5] GNU, "Quagga." [Online]. Available: http://www.quagga.net
[6] A. Bianco, R. Birke, J. M. Finochietto, G. Galante, M. Mellia, M. Prashant, and F. Neri, "Click vs. Linux: Two efficient open-source IP network stacks for software routers," in *Proc. of the IEEE Workshop on High Performance Switching and Routing (HPSR 2005)*, Hong Kong, P.R. China, May 12–14, 2005, pp. 18–23.
[7] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*. Los Alamitos, CA, US: IEEE Computer Society Press, 1997.
[8] C.-S. Chang, D.-S. Lee, and Y.-S. Jou, "Load balanced birkhoff-von neumann switches, part i: one-stage buffering." *Computer Communications*, vol. 25, no. 6, pp. 611–622, 2002.
[9] I. Keslassy and N. McKeown, "Maintaining packet order in two-stage switches," in *Proceedings of IEEE INFOCOM*, New York, NY, US, June 23–27 2002, pp. 1032–1042.
[10] Agilent, "N2X RouterTester 900." [Online]. Available: http://advanced.comms.agilent.com/n2x
[11] "BoraBora: Building Open Router Architectures Based on Router Aggregation." [Online]. Available: http://www.tlc-networks.polito.it/borabora

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.
Authorized licensed use limited to: Politecnico di Torino. Downloaded on April 27,2010 at 07:24:26 UTC from IEEE Xplore. Restrictions apply.