# RFC 2544 Performance Evaluation and Internal Measurements for a Linux Based Open Router

Raffaele Bolla, Roberto Bruschi
DIST - Department of Communications, Computer and Systems Science
University of Genoa
Via Opera Pia 13, 16145 Genova, Italy
raffaele.bolla, roberto.bruschi@unige.it

***Abstract*—Recent technological advances give a good chance to do something really effective in the field of open Internet equipments, also called Open Routers (ORs). Some initiatives have been activated since the last few years to investigate the OR and related issues. But despite these activities, large interesting areas still require a deeper investigation. This work tries to give a contribution by reporting the results of an in-depth activity of optimization and testing realized on a PC Open Router architecture based on Linux software and COTS hardware. The main target approached in this paper has been the forwarding performance evaluation of different OR Linux-based SW architectures. This analysis has been performed with both external (throughput and latencies) and internal (profiling) measurements. In particular, for what concerns the external measurements, a set of RFC2544 compliant tests has been proposed and analyzed.**

***Keywords-Open Router; RFC 2544; IP forwarding.***

## I. INTRODUCTION

The Internet technology has been developed in an open environment; all Internet related protocols, architectures and structures are publicly created and described. For this reason, in principle, everyone can "easily" develop an Internet equipment (e.g., a router). On the contrary, and in some sense quite surprising, most of the professional equipments are realized in a very "closed" way. Generally speaking, this does not appear so strange: it is a clear attempt to protect the industrial investment. But sometime the "experimental" nature of Internet and its diffusion in many contests suggest a different approach (this is more evident inside the scientific community).

Today, the recent technology advances give a good chance to do something really effective in the field of open Internet equipments, sometime called Open Routers (OR). This possibility comes from the Open Source Operative Systems (OSs) and from the COTS/PC components. The attractiveness of the OR solution can be summarized in multi-vendor availability, low-cost and continuous update/evolution of the basic parts. For what concerns the performance, the PC architecture is a general-purpose and this means that, in principle, it cannot reach the same performance level of custom high-end network equipments. Otherwise, the performance gap might be not so large and anyway more than justified by the cost difference.

Some initiatives have been activated since last few years to investigate the OR and related issues. In the software area, one of the most important initiatives is the Click Modular Router Project [1], which proposes an effective solution for the data plane. In the control plane area two important projects can be cited: Zebra [2] and Xorp [3]. Despite custom developments, also some standard Open Source OSs can give a very effective support to an OR realization; the most relevant OSs in this sense are Linux [4, 5] and FreeBSD. Other activities are focused on HW: [6] and [7] propose a router architecture based on PC cluster, and [8] reports some performance results (in packet transmission and reception) obtained with a PC Linux-based testbed. Some evaluations have been realized also on the network boards, see for example [9]. Other interesting works regarding Linux based OR can be found in [10] and [11], where Bianco et al. report some interesting performance results. This work, carried out during the EURO project [12] participation (like [10] and [11]), tries to give a contribution to the investigation by reporting the results of a large activity of optimization and testing realized on a OR architecture based on Linux SW. We have focused our attention on the testing of packet forwarding functionalities. Our main objective has been the performance evaluation of an optimized OR, with both external (throughput and latencies) and internal (profiling) measurements. With this respect, we have identified a high-end reference PC based HW architecture and Linux kernel 2.5 for the SW data plane, we have optimized this OR structure, defined a test environment and finally realized a complete set of tests with an accurate evaluation of the SW module role in the definition of performance limits.

The paper is organized as in the following. Section II reports the HW and SW details of the proposed OR architecture. In Section III and Section IV, the benchmarking scenario and the performance results are reported. The conclusions are in Section V.

## II. HARDWARE AND SOFTWARE ARCHITECTURE

In the following we have summarized the HW/SW architecture that we have selected to realize the OR. In particular, for concerns the HW, we have to take into account that, during networking operations, the PC internal data path has to use a centralized I/O structure composed by: the I/O bus, the memory channel (both used by DMA to transfer data from network interfaces to RAM and vice versa) and the Front Side Bus (FSB) (used by the CPU with memory channel to access to the RAM during the packet elaboration). So the selection criterions are very fast internal busses and CPUs with high integer computational power. With this goal we have chosen Supermicro X5DL8-GG mainboard, which can support a dual-Xeon system with a dual memory channel and a PCI-X bus at

133MHz with 64 parallel bits. Network interfaces are another critical element, as they can heavily condition the PC Router performance. As reported in [9], the network adapters on the market offer different levels of performance and a different configurability. With this respect, we have selected two different types of adapters with different features and speed: a high performance and configurable Gigabit Ethernet interface, namely Intel PRO 1000 XT Server (PCI-X) [13]; a D-Link DFE-580TX (PCI-2.1) [14] with four Fast Ethernet interfaces.

The OR SW architecture has to provide many different functionalities: the packet forwarding, the control functionalities, the dynamic configuration and the monitoring. We have chosen to study and to analyze a Linux based OR framework, because Linux has a large and sophisticated kernel-integrated network support.

As outlined in [15] and in [5], while all the forwarding functions are realized inside the Linux kernel, the large part of the control and monitoring operations (the signalling protocols like, for example, routing protocols, control protocols, …) are daemons/applications running in user mode. Thus, we have to outline that, unlike most of the high-end commercial network equipments, the forwarding functionalities and the control ones have to share the CPUs in the system. [16] reports a detailed description of how the resource sharing between the control plane and the forwarding process can have effect on the overall performance in different OR configurations (e.g., SMP kernel, single processor kernel, etc.).

The critical element for the IP forwarding is the kernel where all the link, network and transport layer operations are realized[1]. During the last years, the networking support integrated in the Linux kernel has experienced many structural and refining developments. For these reasons, we have chosen to use a last generation Linux kernel, more in particular a 2.5 version. Today, he forwarding mechanism of all Linux kernels is fundamentally composed by a chain of three different modules: a "reception API" that handles the packet reception (NAPI), a module that carries out the IP layer elaboration and, finally, a "transmission API" that manages the forwarding operations to the egress network interfaces. The NAPI architecture [17] has been explicitly created to increase the system scalability, as it can handle network interface requests with a interrupt moderation mechanism, which allows to adaptively switch from a classical interrupt management of the network interfaces to a polling one. About the network structure in 2.5 kernels, we have to note that all the kernel code is structured with a zero-copy statement [18]: to avoid unnecessary and onerous memory transfer operations of packets, they are left to reside in the memory locations used by the DMA-engines of ingress network interfaces, and each following operation is performed by using a sort of pointer to the packet and to its key fields, called descriptor or *sk_buff*. These descriptors are effectively composed by pointers to the different fields of the headers contained in the associated packets. A descriptor is immediately allocated and associated to each received packet in the sub-IP layer, subsequently used in every networking operation, and finally deallocated when the network interface signals the successful transmission.

[1] To be more detailed, some layer 2 operations are directly realized in the network interface drivers

Another interesting characteristic of the in NAPI kernels (introduced to reduce performance deteriorations due to CPU concurrency) is the Symmetric Multi-Processors (SMP) support that may assign the management of each network interface to a single CPU for both the transmission and reception functionalities.

The whole networking kernel architecture is quite complex and it has many aspects and parameters that should be tuned for a system optimization. In particular, our optimized NAPI kernel image ([11], [15] and [16]) includes the descriptor recycling patch [19] (used and described also in [11]) and the 5.2.39-k2 version of e1000 driver, and it has been configured with the following optimized values: the Rx and Tx ring buffers have been set to 256 descriptors; the Rx interrupt generation has not been limited; the *qdisc* size for all the adapters has been dimensioned equal to 20,000 descriptors; the scheduler clock frequency has been fixed to 100Hz.

## III. Benchmarking Scenario

To benchmark the OR forwarding performance, we have used a professional equipment, namely Agilent N2X Router Tester [20], which allows to obtain throughput and latency measurements with very high availability and accuracy levels (i.e., the minimum guaranteed timestamp resolution is 10 ns). Moreover, with two dual Gigabit Ethernet cards and one 16 Fast Ethernet card, it allows us to analyze the OR behaviour with a large number of Fast and Gigabit Ethernet interfaces.

To better support the performance analysis and to identify the OR bottlenecks, we have also performed some internal measurements by using specific SW tools (called profilers) placed inside the OR, which are able to trace the percentage of CPU utilization for each SW modules running on the node. The problem is that many of these profilers require a relevant computational effort that perturbs the system performance. We have experimentally verified that one of the best is Oprofile [21], an open source tool that realizes a continuous monitoring of system dynamics with a frequent and quite regular sampling of CPU HW registers. Oprofile allows the effective evaluation of the CPU utilization of both each SW application and each single kernel function with a very low computational overhead.

About the benchmarking scenario, we have chosen to start by defining a reasonable set of test setups (with an increasing level of complexity), and, for each selected setup, to apply some of the tests defined in the RFC 2544 [22]. In particular, we have chosen to perform these activities by using both a core and an edge router configuration: the first one is composed by few high-speed (Gigabit Ethernet) network interfaces, while the last is composed by a high-speed gateway interface and a large number of Fast Ethernet cards, which collect the traffic from the access networks.


Figure 1. Benchmarking setups.

More in detail, we have performed our tests by using the following setups (see Figure 1. ): Setup A: a single mono directional flow crosses the OR from a Gigabit port to another one; Setup B: a full-meshed (and full-duplex) traffic matrix applied on 4 Gigabit Ethernet ports; Setup C: a full-meshed (and full-duplex) traffic matrix applied on 1 Gigabit Ethernet

port and 12 Fast Ethernet interfaces. In particular, each OR forwarding benchmarking session is essentially composed by three test sets namely:

1. Throughput and latency: this test set is performed by using CBR traffic flows, composed by fixed size datagrams, to obtain: a) the maximum effective throughput (in Kpps and in % with respect to the theoretical value) versus different IP datagram sizes; b) the average, maximum and minimum latencies versus different IP datagram sizes;
2. Back-to-back: these tests are fulfilled by using bursty traffic flows and by changing both the burst dimension (i.e., the number of the packets composing the burst) and the datagram size. The main results are: a) zero loss burst length versus different IP datagram sizes; b) average, maximum and minimum latencies versus different sizes of IP datagram composing the burst ("zero loss burst length" is the maximum number of packets transmitted with minimum inter-frame gaps that the System Under Test (SUT) can handle without any loss).
3. Loss Rate: this kind of test is fulfilled by using CBR traffic flows with different offered loads and IP datagram sizes; the obtainable results can be summarized in throughput versus both offered load and IP datagram sizes.

Note that all these tests have been performed by using different IP datagram sizes (i.e., 40, 64, 128, 256, 512, 1024 and 1500 bytes) and both CBR and bursty traffic flows.

## IV. Numerical Results

In this Section, a selection of the numerical results, obtained with the benchmarking techniques and setups described in Section IV, are reported. We have decided to compare different 2.5.75 Linux kernel configurations and a Click Modular Router. In particular, we have decided to test the following versions of the 2.5.75 Linux kernel: dual-processor 2.5.75 standard kernel (it is a standard NAPI kernel version similar to the previous one but with SMP support); single-processor 2.5.75 optimized kernel (it is a version based on the standard one with single processor support that includes the descriptor recycling patch). Note that we have chosen to not take into account the SMP versions both of the optimized Linux kernel and of the Click Modular Router, as these versions lack of a minimum acceptable stability level.

### A. Setup A numerical results

In the first benchmarking session, we have performed the RFC 2544 tests by using the setup A (see Figure 1. ) with both the single-processor 2.5.75 optimized kernel and Click. As we can observe in Figs. 2, 3 and 4, which report the numerical results of the throughput and latency tests, both the SW architectures cannot achieve the maximum theoretic throughput in presence of small datagram sizes. As demonstrated by the profiling measurements reported in Fig. 5, obtained with the single processor optimized 2.5 kernel and with 64 Bytes sized datagrams, this effect is caused by the CPU that limits the maximum forwarding rate of kernel to about 700 Kpps. In fact, even if the CPU idle goes to zero in correspondence of an offered load equal to 300 Kpps, the CPU occupancies of all the most important function sets appear to adapt their contributes till 700 Kpps; after this point their % contributions to the CPU utilization remains almost constant. More in particular, Fig. 5 shows that the computational weight of memory management operations (like *sk_buff* allocations) is substantially limited, thanks to descriptor recycling patch, to less than the 1%.

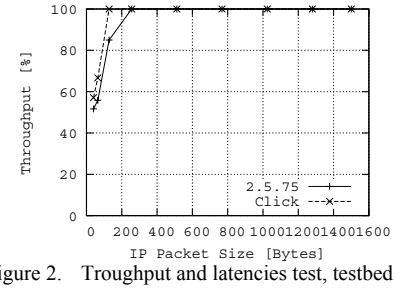


Figure 2. Troughput and latencies test, testbed A: effective throughput results.

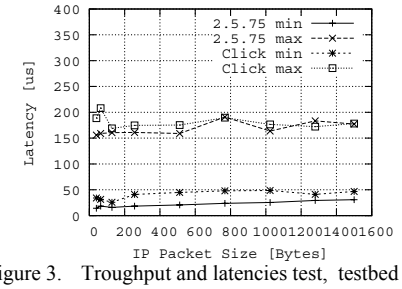


Figure 3. Troughput and latencies test, testbed A: minimum and maximum latencies.

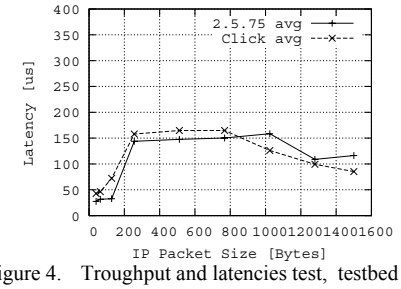


Figure 4. Troughput and latencies test, testbed A: average lantecies.

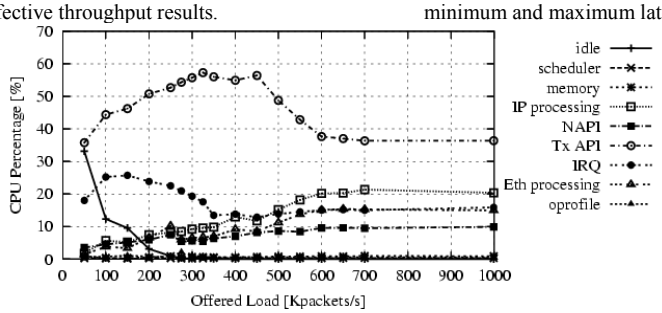


Figure 5. Profiling results of the optimized Linux kernel obtained with the testbed setup A

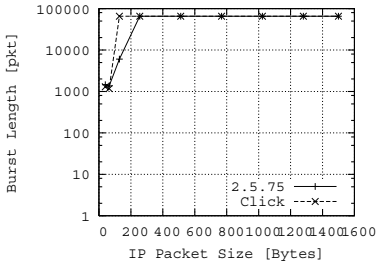


Figure 6. Back-to-back test, testbed A: maximum zero loss burst lengths.

Both the TxAPI (the most onerous operation set that takes at minimum about the 35% of the overall resources) and interrupt management operations apparently have strange and somehow similar behaviours: their CPU utilization level, after an initial growth, decreases with the increase of the input rate. This behaviour is mostly due to two different reasons related with the packet grouping effect in the Tx and the RxAPI: when the ingress packet rate raises, NAPI tends to moderate the Rx interrupt rate by changing its behaviour from a interrupt-like mechanism to a polling one (so we have a first interrupt number reduction), while TxAPI, in the same condition, can better exploit packet grouping mechanism by sending more packets at time (and then the number of interrupts for successful Tx confirmations decreases). About IP and Ethernet

processing operations, we can note that they increase their CPU % utilizations in an almost linear way with respect to the number of the forwarded packets per second. Considerations similar to the previous ones can be done also for what concerns Click.

TABLE I. BACK-TO-BACK TEST, TESTBED A: LATENCY VALUES.

| PktLength [Byte] | optimized 2.5.75 Kernel Latency Min [us] | Average [us] | Max [us] | Click Latency Min [us] | Average [us] | Max [us] |
|---|---|---|---|---|---|---|
| 40 | 16.16 | 960.08 | 1621.47 | 23.47 | 1165.53 | 1693.64 |
| 64 | 14.95 | 929.27 | 1463.02 | 23.52 | 1007.42 | 1580.74 |
| 128 | 16.04 | 469.9 | 925.93 | 19.34 | 54.88 | 53.45 |
| 256 | 16.01 | 51.65 | 58.84 | 22.49 | 52.62 | 47.67 |
| 512 | 18.95 | 54.96 | 61.51 | 20.72 | 62.92 | 59.95 |
| 768 | 23.35 | 100.76 | 164.56 | 22.85 | 116.61 | 155.59 |
| 1024 | 25.31 | 123.68 | 164.21 | 32.02 | 128.85 | 154.72 |
| 1280 | 28.6 | 143.43 | 166.46 | 24.77 | 151.81 | 178.45 |
| 1500 | 30.38 | 142.22 | 163.63 | 32.01 | 154.79 | 181.43 |

For the same reasons, as shown in Figs. 3 and 4, the receive mechanism embedded in Click introduces higher packet latencies. According to the previous results, also the back-to-back tests, reported in Fig. 6 and Table I, show that the optimized 2.5.75 Linux kernel and Click continue to suffer small sized datagrams. In fact, while using 256 Bytes or higher sized datagrams the measured zero-loss burst length is quite close to the maximum burst length used in the performed tests, it appears to be heavily limited in presence of 40, 64 and, only for what concerns the Linux kernel, 128 Bytes sized packets.

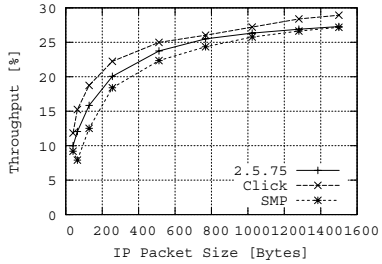


Figure 7. Troughput and latencies test, Setup B: effective throughput results.

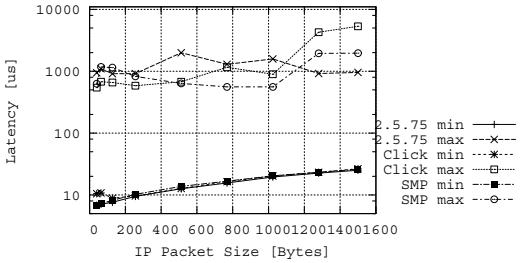


Figure 8. Troughput and latencies test, testbed B: minimum and maximum latencies for all the three SW architectures.

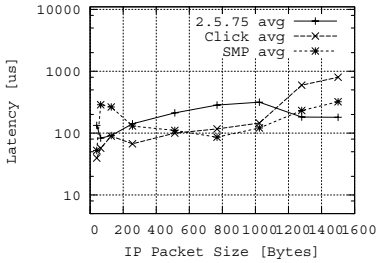


Figure 9. Troughput and latencies test, results for testbed B: average lantecies.

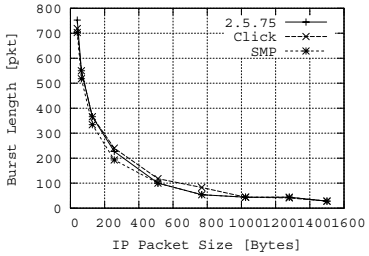


Figure 10. Back-to-back test, testbed B: maximum zero loss burst lengths.

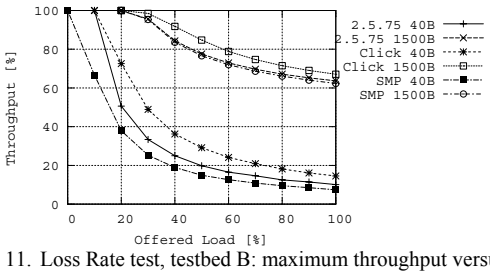


Figure 11. Loss Rate test, testbed B: maximum throughput versus both offered load and IP datagram sizes

Observing these lasts, we can note how the SMP kernel with short datagrams continues to suffer memory concurrency problems lowing on the OR performance and considerably increasing both the average and the maximum latency values. In Fig 16 and Tables III, which report the back-to-back test results, all the three OR architectures achieve a similar zero-loss burst length, while Click reaches very high average and maximum latencies with respect to the single-processor and SMP kernels. The loss-rate results in Fig. 17 highlight the performance decay of the SMP kernel, while a fairly similar behaviour is achieved by the other two architectures. Moreover, like in previous benchmarking session, the maximum forwarding rate of each Gigabit network interface is limited to about 600/650 Mbps.

Apart from the single 128 Bytes case, where NAPI starts to suffer the computational bottleneck while Click continues to have a forwarding rate very close to the theoretic one; the Linux kernel provides a better support to bursty traffic than Click resulting in both larger zero loss burst lengths and smaller associated latency times. In Fig. 7, the loss rate test results are finally reported.

### *B. Setup B numerical results*

The second benchmarking session, the three considered SW architectures have been tested in presence of four Gigabit interfaces with a fullmeshed traffic matrix (Fig. 1). By analyzing the maximum effective throughput values in Fig. 13, we can note that Click has better performance with respect to the Linux kernels, while the 1CPU kernel provides maximum forwarding rates considerably larger than the SMP version. In fact, the SMP kernel tries to share the computational load of the incoming traffic among the CPUs, resulting in a almost static assignment of each CPU to two specific network interfaces. As, in presence of a full-meshed traffic matrix, about the half of the forwarded packets cross the OR between two interfaces managed by different CPUs, this causes a performance lack due to memory concurrency problems [15, 16]. Figs. 14 and 15 show the minimum, the maximum and the average latency values obtained during this test set.

TABLE II. BACK-TO-BACK TEST, TESTBED B: LATENCY VALUES

| Pkt Length [Byte] | optimized 2.575 Kernel Latency [us] Min | Average | Max | Click Latency [us] Min | Average | Max | 2.5.75 SMP Kernel Latency [us] Min | Average | Max |
|---|---|---|---|---|---|---|---|---|---|
| 40 | 92.1 | 2424.3 | 5040.3 | 73.5 | 6827.0 | 15804.9 | 74.8 | 3156.8 | 6164.2 |
| 64 | 131.3 | 1691.7 | 3285.1 | 176.7 | 6437.6 | 16651.1 | 66.5 | 2567.5 | 5140.6 |
| 128 | 98.1 | 1281.0 | 2865.9 | 60.2 | 3333.8 | 9482.1 | 77.1 | 1675.1 | 3161.6 |
| 256 | 19.6 | 915.9 | 2494.6 | 16.7 | 1388.9 | 3972.2 | 44.1 | 790.8 | 1702.8 |
| 512 | 23.9 | 666.9 | 2138.9 | 15.9 | 649.2 | 2119.3 | 23.2 | 815.8 | 2189.0 |
| 768 | 22.3 | 571.3 | 2079.7 | 22.5 | 543.7 | 2002.6 | 23.6 | 737.3 | 2193.0 |
| 1024 | 22.0 | 353.7 | 1232.2 | 36.3 | 382.2 | 1312.7 | 30.0 | 411.7 | 1276.8 |
| 1280 | 25.9 | 436.4 | 1525.4 | 34.6 | 443.0 | 1460 | 29.8 | 447.7 | 1469.5 |
| 1500 | 27.4 | 469.5 | 1696.7 | 36.7 | 457.5 | 1525.7 | 30.0 | 482.6 | 1719.6 |

### *C. Setup C numerical results*

In the last benchmarking session, we have applied the Setup C, which provides a full-meshed traffic matrix between one Gigabit Ethernet and 12 Fast Ethernet interfaces, to the single-processor Linux kernel and to the SMP version. We have not

used Click in this last test, as, at the moment and for this SW architecture, there is no drivers with polling support for the D-Link interfaces. By analyzing the throughput and latency results in Figs. 18, 19 and 20, we can note how, with many interfaces and a full-meshed traffic matrix, the SMP kernel version annihilates its performance: the maximum effective throughput is limited to about 2400 packets/s and the corresponding latencies appear clearly larger with respect to the ones obtained with the single processor kernel. However, also the single processor kernel does not support the maximum theoretic rate: it achieves the 10% of the full speed in presence of short sized datagrams and about the 75% for large datagram.

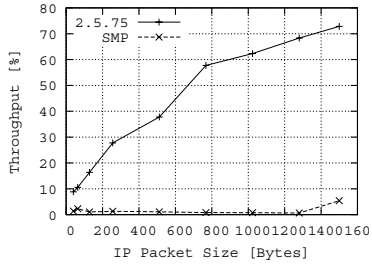


Figure 12. Troughput and latencies test, Setup C: effective throughput results.

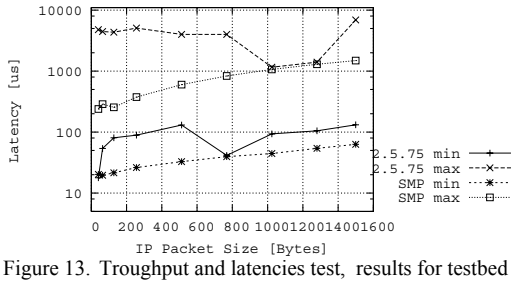


Figure 13. Troughput and latencies test, results for testbed C: minimum and maximum latencies.

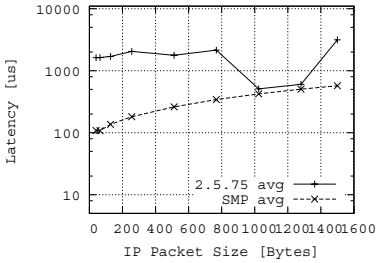


Figure 14. Troughput and latencies test, testbed C: average lantecies for both the Linux kernels.

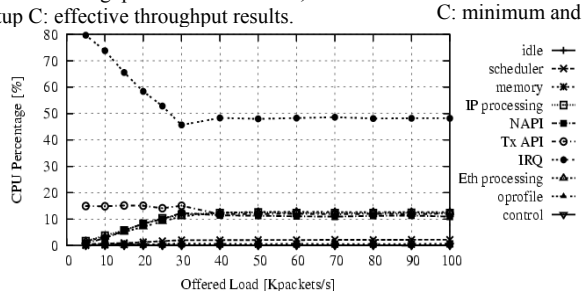


Figure 15. Profiling results obtained by using 12 CBR flows that cross the OR from the Fast Ethernet interfaces to the Gigabit one.

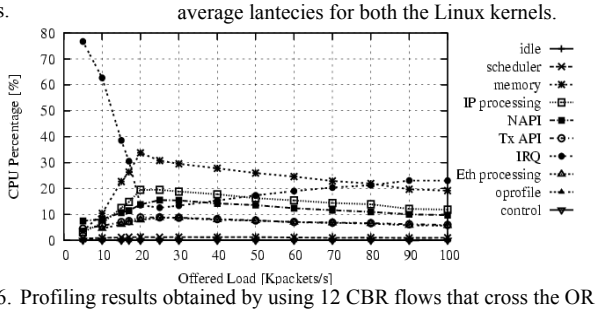


Figure 16. Profiling results obtained by using 12 CBR flows that cross the OR from a Gigabit interface to 12 FastEthernet ones.

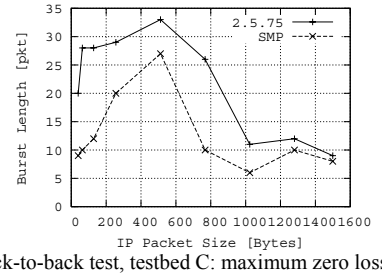


Figure 17. Back-to-back test, testbed C: maximum zero loss burst lengths.

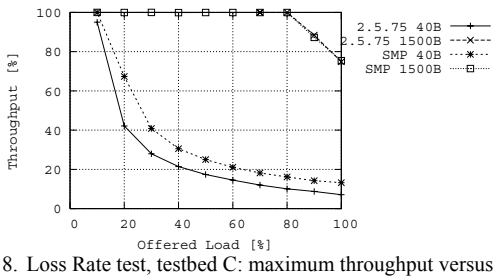


Figure 18. Loss Rate test, testbed C: maximum throughput versus both offered load and IP datagram sizes.

To better understand why the OR does not reach the full-speed with a so high number of interfaces, we have decided to perform several profiling tests. These tests were carried out by using two simple traffic matrixes: the first (Fig. 21) is composed by 12 CBR flows that cross the OR from the Fast Ethernet interfaces to the Gigabit one, while the second (Fig. 22) is still composed by 12 CBR flows that cross the OR with the opposite direction (e.g., from the Gigabit to the Fast Ethernet interfaces). The Figs. 21 and 22 report the profiling results corresponding to the two traffic matrixes. The internal measurements in Fig. 21 highlight that the CPU is overloaded by the IRQ and TX API management operations, which can be retraced to the fact that, during the Tx process, each interface has to signal to the associated driver instance through interrupts the state of transmitting packets and of the Tx ring. By observing again Fig. 21, we can note that the IRQ CPU occupancy decreases up to the 30% of the offered load, and after, while the OR achieve the saturation, it uses constantly about the 50% of the CPU time. The initial decreasing behavior is due to the fact that increasing the traffic offered load, the OR can better exploit packet grouping effects. Passing to Fig. 22, we can note how the presence of traffic incoming from many interfaces rises the computational weights of both the IRQ and the memory management operations. The decreasing behavior of IRQ weight is due to the typical NAPI structure that passes from an IRQ based mechanism to a polling one. The large weight of the memory management is explainable with the fact that the recycling patch is not working with the Fast Ethernet driver. The back-to-back results, reported in Fig 23 and Table IV, show a very particular behavior: even if the single processor kernel can achieve longer zero-loss burst lengths than the SMP kernel, this last appears to assure lower minimum, average and maximum latency values. In the end, Fig. 24 reports the loss rate test results, which, compatibly with the previous obtained results, show that single processor kernel can sustain a larger throughput than the SMP.

TABLE III. BACK-TO-BACK TEST, TESTBED D: LATENCY VALUES.

| | Optimized 2.575 Kernel | | | 2.5.75 SMP Kernel | | |
|---|---|---|---|---|---|---|
| PktLength | Latency [us] | | | Latency [us] | | |
| [Byte] | Min | Average | Max | Min | Average | Max |
| 40 | 285.05 | 1750.89 | 2921.23 | 37.04 | 1382.96 | 2347.89 |
| 64 | 215.5 | 1821.81 | 2892.13 | 38.07 | 1204.43 | 1963.7 |
| 128 | 216.15 | 1847.76 | 3032.22 | 34.52 | 1244.87 | 1984.4 |
| 256 | 61.83 | 1445.15 | 2353.12 | 30.61 | 2082.76 | 3586.6 |
| 512 | 57.73 | 2244.68 | 4333.44 | 32.97 | 908.19 | 1661.18 |
| 768 | 101.78 | 1981.5 | 3497.81 | 50.64 | 1007.27 | 1750.45 |
| 1024 | 108.17 | 1386.19 | 2394.4 | 52.14 | 819.98 | 1642.13 |
| 1280 | 73.15 | 1662.13 | 3029.54 | 58.11 | 981.92 | 1953.62 |
| 1500 | 109.24 | 1149.76 | 2250.78 | 70.92 | 869.36 | 1698.68 |

In order to effectively synthesize and better evaluate the proposed performance results, we report in Figs. 25 and 26 the aggregated[2] maximum values for each used testbed of respectively the effective throughput and the maximum throughput (obtained in the loss rate test). By analyzing Fig. 25, we can note that with many interfaces, the OR achieves values higher than 1 Gbps, and, in particular, that it reaches a maximum values equal to 1.6 Gbps with the testbed D. The values of aggregated maximum throughput, reported in Fig. 26, are obviously larger than the ones in Fig. 25, and highlight that the maximum forwarding rates sustainable by the OR are achieved in setup B with 2,5 Gbps. Moreover, we can note that, while in setup A the maximum theoretical rate is achieved for packet sizes larger than 128, in all the other setups the maximum obtained throughputs reach values not much higher than the half of theoretical values.

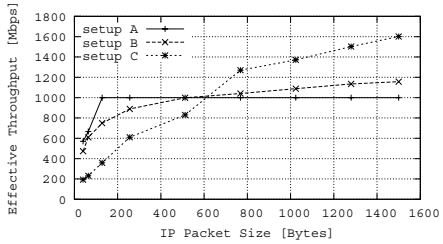

Figure 19. Maximum effective throughput values obtained in the used testbeds.

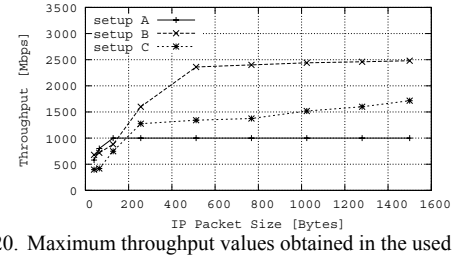

Figure 20. Maximum throughput values obtained in the used testbeds.

## V. CONCLUSIONS

This work tries to give a contribution by reporting the results of a deep activity of optimization and testing realized on a PC Open Router architecture based on Linux SW, and, more in particular, based on Linux kernel. The main objective has been the forwarding performance evaluation of three different architectures, including optimized Linux kernel, Click Modular Router and SMP Linux kernel, both with external and internal (profiling) measurements. External measurement have been performed in a RFC2544 compliant way by using professional equipments. The obtained results show that the OR can achieve a quite interesting performance level by achieving aggregated forwarding rate values equal to about 2.5 Gbps with relative low latencies.

## ACKNOWLEDGMENT

A special thanks to Filippo Cugini and Piero Castoldi of S.S. Anna of Pisa for their kindness and help.

## REFERENCES

[1] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek,"*The Click modular router*", ACM Transactions on Computer Systems 18(3), August 2000, pages 263-297.

[2] Zebra, http://www.zebra.org/.

[3] M. Handley, O. Hodson, E. Kohler, "*XORP: an open platform for network research*", ACM SIGCOMM Computer Communication Review, Vol. 33 Issue 1, Jan 2003, pp. 53-57.

[4] S. Radhakrishnan, "*Linux – Advanced networking overview*", http://qos.ittc.ku.edu/howto.pdf.

[5] M. Rio et al., "*A map of the networking code in Linux kernel 2.4.20*", Technical Report DataTAG-2004-1, FP5/IST DataTAG Project, March 2004.

[6] B. Chen and R. Morris, "*Flexible Control of Parallelism in a Multiprocessor PC Router*", Proceedings of the 2001 USENIX Annual Technical Conference (USENIX '01), Boston, Massachusetts, June 2001.

[7] C. Duret, F. Rischette, J. Lattmann, V. Laspreses, P. Van Heuven, S. Van den Berghe, P. Demeester, "*High Router Flexibility and Performance by Combining Dedicated Lookup Hardware (IFT), off the Shelf Switches and Linux*", Proc. of Second International IFIP-TC6 Networking Conference, Pisa, Italy, May 2002, LNCS 2345, Ed E. Gregori *et al*, Springer-Verlag 2002, pp. 1117-1122.

[8] A. Barczyk, A. Carbone, J.P. Dufey, D. Galli, B. Jost, U. Marconi, N. Neufeld, G. Peco, V. Vagnoni, "*Reliability of datagram transmission on Gigabit Ethernet at full link load*", LHCb technical note, LHCB 2004-030 DAQ, March 2004.

[9] P. Gray, A. Betz, "*Performance Evaluation of Copper-Based Gigabit Ethernet Interfaces*", 27th Annual IEEE Conference on Local Computer Networks (LCN'02), Tampa, Florida, November 2002, pp.679-690.

[10] A. Bianco, R. Birke, D. Bolognesi, J. M. Finochietto, G. Galante, M. Mellia, M.L.N.P.P. Prashant, Fabio Neri, "*Click vs. Linux: Two Efficient Open-Source IP Network Stacks for Software Routers*", HPSR 2005 (IEEE Workshop on High Performance Switching and Routing), Hong Kong, May 12-14, 2005

[11] A. Bianco, J. M. Finochietto, G. Galante, M. Mellia, F. Neri, "*Open-Source PC-Based Software Routers: a Viable Approach to High-Performance Packet Switching*", Third International Workshop on QoS in Multiservice IP Networks, Catania, Italy, Feb 2005

[12] EURO project, http://www.diit.unict.it/euro.

[13] The Intel PRO 1000 XT Server Adapter, http://www.intel.com /network/connectivity/products/pro1000xt.htm.

[14] The D-Link DFE-580TX quad network adapter, http://support. dlink.com/products/view.asp?productid=DFE%2D580TX#.

[15] R. Bolla, R. Bruschi, "*A high-end Linux based Open Router for IP QoS networks: tuning and performance analysis with internal (profiling) and external measurement tools of the packet forwarding capabilities*", Proc. of the 3rd International Workshop on Internet Performance, Simulation, Monitoring and Measurements (IPS MoMe 2005), Warsaw, Poland, Mar. 2005.

[16] R. Bolla, R. Bruschi, "*IP forwarding Performance Analysis in presence of Control Plane Functionalities in a PC-based Open Router*", Proc. of the 2005 Tyrrhenian International Workshop on Digital Communications (TIWDC 2005), Sorrento, Italy, Jun. 2005, and in F. Davoli, S. Palazzo, S. Zappatore, Eds., "Distributed Cooperative Laboratories: Networking, Instrumentation, and Measuments", Springer, Norwell, MA, 2006, pp. 143-158.

[17] J. H. Salim, R. Olsson, A. Kuznetsov, "*Beyond Softnet*", Proceedings of the 5th annual linux Showcase & Conference, November 2001, Oakland California.

[18] A. Cox "*Network Buffers and Memory Management*" *Linux Journal*, Oct. 1996, http://www2.linuxjournal.com/ lj−issues/issue30/1312.html.

[19] The descriptor recycling patch, ftp://robur.slu.se/pub/Linux/net-development/skb_recycling/.

[20] The Agilent N2X Router Tester, http://advanced.comms.agilent.com /n2x/products/index.htm

[21] Oprofile, http://oprofile.sourceforge.net/news/.

[22] RFC 2544, http://www.faqs.org/rfcs/rfc2544.html.

[2] With "aggregated" we intend the sum of the forwarding rates of all the OR network interfaces.