# Distributed PC Based Routers: Bottleneck Analysis and Architecture Proposal

A. J. Khan, IIT Bombay
R. Birke, Politecnico Di Torino
D. Manjunath, IIT Bombay
A. Sahoo, IIT Bombay
A. Bianco, Politecnico Di Torino

*Abstract*— **Recent research in the different functional areas of modern routers have made proposals that can greatly increase the efficiency of these machines. Most of these proposals can be implemented quickly and often efficiently in software. We wish to use personal computers as forwarders in a network to utilize the advances made by researchers. We therefore examine the ability of a personal computer to act as a router. We analyze the performance of a single general purpose computer and show that I/O is the primary bottleneck. We then study the performance of distributed router composed of multiple general purpose computers. We study the performance of a star topology and through experimental results we show that although its performance is good, it lacks flexibility in its design. We compare it with a multistage architecture. We conclude with a proposal for an architecture that provides us with a forwarder that is both flexible and scalable.**

## I. Introduction

The changing requirements in routers demands flexibility in its design. The functions performed by a router at different time scales have varied over the years. Current routers are the fourth generation in their evolution [1]. Modern routers are no longer passive forwarders of traffic. The hardware and software in routers have been creatively designed to keep up with the new requirements imposed on them [2]. These include supporting firewalls, encryption capabilities, QoS, per-user forwarding rules, very deep packet inspection, Virtual Private Networks, Network Address Translation and more. These advances have been in the architecture, the algorithms and the functionalities offered by the routers.

Most of these research proposals can be implemented in software and hardware. We believe that implementation in software on a personal computer (PC) is easier and sometimes quicker because tremendous support tools are available. It is therefore natural to ask whether a general purpose computer can act as a networking element—specifically a router. We already have a large body of software that enables a general purpose computer to act as a router. The next step is to identify the performance characteristics of this PC based router.

A router consists of three distinct planes of operation—control, data and management. Each plane works at a different time scale and hence requires a different level of performance. The most performance demanding plane is the data plane where the timescale is at the bit level. A performance evaluation of a PC in the data plane should give us a good idea of the capabilities of a personal computer. We have, therefore, restricted ourselves only to the data plane for the present. Our study covers non-commercial publicly described hardware and software. Previous work on this [3] has concluded that a high end general purpose computer is limited by its I/O bus. We extend this work to account for different forwarding table sizes and different number of flows. We start with a comparison of the Linux kernel's [4] performance with Click [5] software.

We observe like Bianco et al. [3] that a single general purpose computer is not sufficiently powerful to handle even one Gigabit per second traffic. Therefore, we propose a distributed design that uses multiple PCs. We evaluate the performance of one such design where the computers are connected to each other in a star topology through a high speed switch. This set of computers forms a single logical router and performs much better than the single PC setup. However, we notice that because of it's symmetric design, it is less than ideal in certain cases.

We therefore look into another proposed design [6] which is a two-stage architecture. The first stage or the front-end acts like a layer 2 load balancer. The second stage or the back-end consists of the layer 3 forwarders. We examine the performance characteristics of this architecture and the flexibility it offers for the implementation of certain research proposals.

The remainder of the paper is organized as follows. In Section II we have an overview of some of these research proposals. Section III discusses the experimental performance of the single computer setup for both Click [5] and Linux. Section IV discusses the experimental performance results of one distributed architecture. Section V discusses the multistage architecture proposal for a router. We offer some conclusions in Section VI.

## II. Recent Research Proposals In Routers

Some researchers have proposed adding many new functions on a router [7]. Some of these also propose a fresh approach in the internal design as routers are not just passive forwarders of traffic anymore. Deeper inspection of packets may allow a router to provide better functionality including security at the edge of a network. Several approaches have been suggested [8], [9].

For packet processing, instead of ASICs, network processor (NP) based routers were designed because they offer programming flexibility and yet are more specific in their abilities to process packets than general purpose CPUs [10]. Parallel processing of packets has been attempted in routers. Most modern routers perform the forwarding function for most packets on the line card itself [1]. There are proposals to take advantage of pipelining to process packets in parallel.

It was previously argued for the end-to-end design philosophy in the network [11] and for separation of functions within the device itself [12]. With new

978-1-4244-1982-1/08/$25.00 ©2008 IEEE 

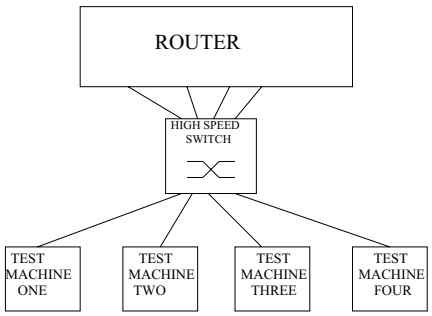


Fig. 1. The router and machine interconnections

services being demanded of the routers, new proposals for its software architecture have been put forward [13]. Principles of software engineering are used for better performance. A study of a few different architectures [14] concludes that modularization creates a flexible and extensible router. Linux in the data plane, Quagga and XORP [15] in the control plane, separate the different routing protocols into discrete portions of software. Modularization allows easier decoupling which in turn allows flexibility in design. So far, the software of the control and data planes have been tightly coupled. An argument to separate the control plane (CP) from the data plane (DP) of a router has been proposed by many researchers [16], [17]. Rexford et al. claim that this approach improves reacheability, performance, reliability and security of the network. General purpose computers are used as forwarders. Click [5], a modular software router has been shown to be an impressive alternative on a PC. A router that that used the DEC Alpha as the main forwarder ran hand written assembly code [18].

In the control plane, a router runs distributed algorithms to compute a forwarding table. In the data plane, the forwarding function looks up the forwarding table to determine the next hop of a packet and sends that packet to the next hop. New algorithms and extensions are continuously proposed to improve operations in both these areas [19], [20]. A good survey paper [21] discusses most of the common approaches to packet classification. Hash tables, aggregated bit vectors and decision trees are some techniques. There are some other equally important algorithms in the areas of scheduling, reliability and measurement.

## III. Single Computer Architecture

We first start with a simple base design, i.e. a *single* personal computer (PC), and identify the limitations inherent in it. We wish to identify limitations in the processing capability, the I/O capability and the software of this machine. To achieve this, we run a series of experiments that stress these aspects of this PC. We used Gigabit Ethernet as our link layer and UDP/IP packets. Since the router is to be stress tested, we have injected fixed size packets as a stream of constant bit rate (CBR) traffic. We varied the bit rate of the injected trafffic and observed the system's response.

As shown in Fig. 1, the Linux router is connected to four test machines (TMs). The test machines are used

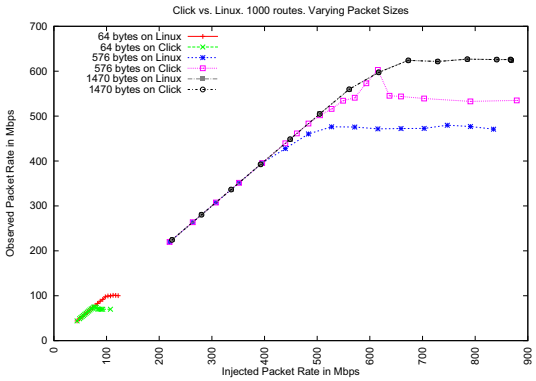


Fig. 2. Click vs. Linux

to source and sink the traffic. There are two source machines and two sinks. Only the router performs layer 3 routing. The TMs and the router are interconnected with a layer 2 Gigabit switch.

Appendices A.1 and A.2 list the details about the computer hardware. The operating system details are mentioned in Appendix B.2 and the tools used in our experiments are detailed in Appendix B.

We first try to identify software that will perform the forwarding in the router. We compare two open source software: Linux and Click.

### A. Performance of Click vs. Linux

Click [5] is a modular, software router for PCs. Linux is the kernel of the GNU/Linux operating system which has built-in ability to act as a router. We compare the performance of Click with Linux's routing ability. For our experiments, we configured Click to adhere to the RFC 1812 standard for the forwarding path.

In our experiments, the tests were run for three different packet sizes (64, 576 and 1470 bytes), for three different number of flows (one, two and many) and for two different forwarding table (FT) sizes (10 and 1000 entries). In this paper, we are reporting the results only for the case of FT size 1000 and many flows. From the graphs in Fig. 2, we note that Click mostly outperforms Linux in terms of forwarding ability for different packet sizes. Click continues to outperform Linux with increasing FT size (which is not shown here).

Thus, the better choice of software for a PC based forwarder is Click. Click unfortunately, did not, as of at the time of our experiments, support large forwarding tables in the kernel mode of operation. Therefore, we used the Linux kernel networking stack for all our remaining experiments.

### B. Experiment Results for a Single Computer Router

A router processes every incoming packet and also decides its next hop. Thus, the processing elements inside the router can be a bottleneck for packet processing rate. Our experiments tested the throughput and therefore the packet processing rate capability of the PC router.

We used three different parameters. The size of the packets, the number of flows (source and destination

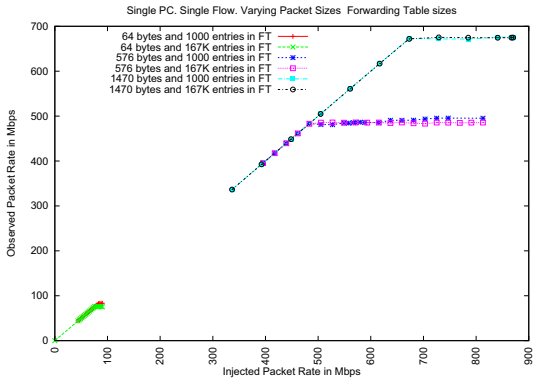


Fig. 3. Single flow on a PC router with FT sizes of 1K and 167K

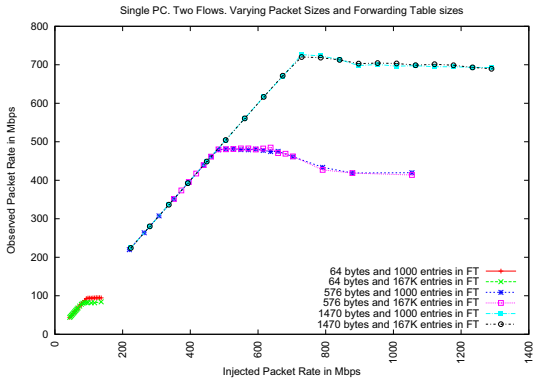


Fig. 4. Two flows on a PC router with FT sizes of 1K and 167K

pairs) and forwarding table size. For packet size, we used three values: 64, 576 and 1470 bytes. Minimum packet size stresses processing elements and hence 64 was chosen. The average packet size on the Internet [22] is acknowledged to be 576 bytes. The maximum packet size stresses the I/O hardware and hence 1470 bytes were chosen since we use Ethernet for our link layer.For number of flows, we chose one, two and many. For one flow, one source computer sent packets destined to the same destination sink computer. This tested the caching mechanism of the router. For two flows, the first source sent packets to the first sink only and the second source sent packets to the second sink only. This tested the software switching and caching mechanism for all links together. For many flows, each source generated random destination IP addresses. Thus successive packets in this scenario could end at different sinks. This test simulates a real world router living away from the edge of a network. Lastly, four different firwarding table sizes were chosen—10, 1000, 10000 and 167000 entries. These entries were taken from a core router on the Internet. The average of 12 readings is reported.

The graphs are plotted to show the input rate and the actual throughput at the router. The graphs compare for the different flow scenarios as well as for different packet sizes.

*1) For a single flow:* We first consider the case for packets in a single flow. Fig. 3 shows the results from our experiments. We have a single flow going from Test Machine 3 to Test Machine 1 via the router. Note that the NIC receiving the packets on the router *is NAPI enabled.* We have shown the results only for FT sizes 1000 and 167000. We make a couple of observations.

First, the packet generation source machine itself is a bottleneck. This can be seen from the fact that the packet injection rate does not increase after a point.

Second, even with a reduced packet input rate (as packet size increased) the Gigabit link remained unsaturated. This despite that, the processor can process almost three times more packets per second (as seen in the 64 bytes case). Clearly, the I/O bus on the PC router and the source is restricting the throughput.

*2) For Two Flows:* We next consider the case for packets in two flows. Here, the first flow is from Test Machine 3 to Test Machine 1. The second flow is from Test Machine 4 to Test Machine 2. We make some observations.

Comparing Fig. 4 with Fig. 3, we observe that the throughput for two flows has increased to a small extent. We noticed during our tests that the router would forward an unequal number of packets from the two flows. This was because one of the reception NICs on the router was using NAPI and hence the scheduling mechanism favored it. The other did not use NAPI leading to possible IRQ thrashing.

We note that the throughput of the router drops after a point. The additional load on the router as it switches between the two flows causes work to be lost as packets get dropped. Processing is thus a bottleneck. We also observe that for 576 byte packets, the drop is more than that for minimum or maximum sized packets. We believe the reason for this behavior is the high cost of bus arbitration mechanism on the PCI bus for average sized packets.

*3) For No Flows i.e. randomly destined packets:* Lastly, we consider the case for no flows wherein, every successive packet coming into the router, has a different destination address. We make two interesting observations from Fig. 5.

First, after a certain packet injection rate there is a sharp drop in the packet processing capability of the router. As the injection rate increases, the router's queue length increases. Sometimes, the packet does not get fetched from the NIC and it is overwritten. These appear as overruns in the NIC statistics. Secondly, as the FT size increases, the packet processing rate drops further. For both of these observations, we believe, route lookup time is the main bottleneck.

### C. Observations for the Single Computer

- **Forwarding Table Size Affects Performance**: From Fig. 5, we observe that the FT size makes a substantial impact on the forwarding rate and can be a bottleneck.
- **Line Rates Not Reached**: It is observed from Fig. 3 and Fig. 4 that the router's processor does not reach its packet processing capacity for larger packet sizes. Thus, the I/O bus is one bottleneck.

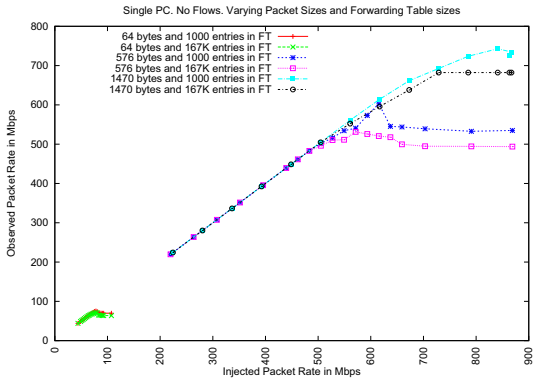


Fig. 5. Many flows on a PC router with FT sizes of 1K and 167K

- **Packet Drops at NICs**: As the injection rate increases into multiple NICs, a large number of overruns are observed in the NICs. This means that the CPU can also be a bottleneck at these values.
- **Plateau Regions**: After a point, the router throughput becomes constant or decreases linearly with a small slope and we see a plateau. As packet size increases, this happens earlier. For 64-byte packets, we did not reach the end of the plateau due to the source bottleneck but we did in the case of the larger packet sizes.

## IV. A Distributed Architecture

The observations in Sec. III-C naturally led us to consider a router made up of a distributed set of PCs. A distributed architecture should allow higher performance. In a distributed architecture, a PCr would now act like a forwarder and interconnected with other PCs similar to itself. All the PCs together act as a single logical router. Therefore, we next address the hardware and software design for our distributed architecture.

There are several basic topologies for interconnection of the computers in our distributed setup—bus, ring, star, mesh etc. A bus topology does not make sense as it has obvious inefficiencies in communication between pairs of nodes in the set. A ring topology has two obvious disadvantages. Not only will a packet have to traverse the network stacks of multiple computers unnecessarily, it fails to provide adequate service if even a single computer goes down. A mesh topology requires $n^2$ interconnections which will quickly cause scaling issues. Thus, we are left with the star topology.

A look at most modern router architecture reveals that each line card in the router is capable of performing forwarding. In the same way, each of our PCs act as forwarders akin to a line card of a router. In the commercial router design, the line cards are connected by a very high speed switching fabric. Similarly, our PCs are interconnected to each other with a high speed switch. Thus, as in Fig. 6, we use a star topology for our architecture and consider the impact on our forwarding performance. The hardware details are listed in Appendix A.2.

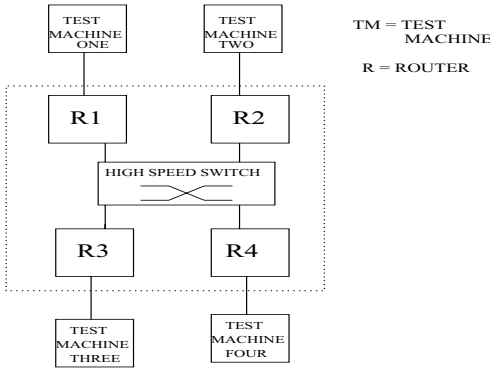


Fig. 6. Distributed Setup. Star Topology

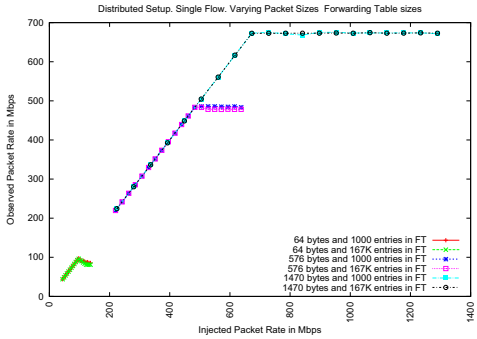


Fig. 7. Single flow on a distributed setup with FT sizes of 1K and 167K

Four PCs act as forwarders—R1, R2, R3 and R4. The dotted box around the four computers is to show that these computers should act as a single logical router. Each forwarder has two NIC cards. One NIC is used for the interconnection amongst the forwarders. The other NIC connects to a test machine directly.

The PCs run software as in Section B.2 and Section B. We performed the same set of experiments as in Section III-B. The two sources sent their packets directly to R3 and R4. The two sinks were connected to R1 and R2. We configured router R3 to forward packets to R1 by default. Similarly, router R4 was configured to forward packets to router R2 by default. For non default entries, R3 sends the packets to R2 and R4 forwards the packets to R1. This prevents the traffic to sink on only one interface in a forwarder during the many flows scenario test.

For the case of single flow, we see from Fig. 7 that the performance is about the same as that of the case for the single computer in Fig. 3.

We also observe that curves in the distributed case are smoother. We assume this to be so because the flow passes through the network stack of two computers.

For the case of two flows, comparing Fig. 8 and Fig. 4, we see almost a doubling in throughput. This is because the two flows do not mix anywhere.

In the case for many flows, since the packets have random destination addresses, the flows from the two sources mix and cross over and hence a more real picture

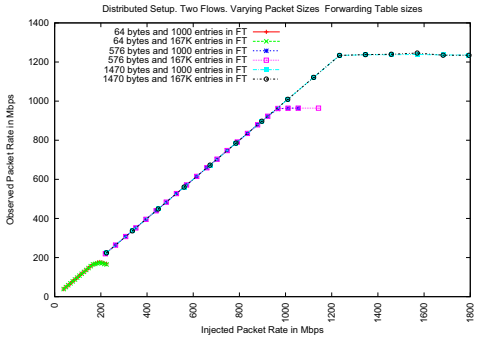


Fig. 8. Two flows on a distributed setup with FT sizes of 1K and 167K

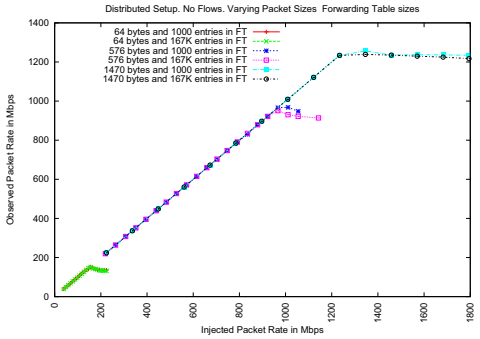


Fig. 9. Many flows on a distributed setup with FT sizes of 1K and 167K

emerges. A comparison of Fig. 9 and Fig. 5 shows that the performance drop is along similar lines. However, in the distributed case, the performance drop is very noticeable for small packet size and not as much for larger packet sizes.

### A. Observations

- **Increased throughput**: The throughput is almost double for every case with more than one flow. Thus, the I/O bottleneck of a single PC can be overcome by using a distributed setup.
- **Line Rates still not achievable**: This is expected as the line rates are still bound by the limitations of the PC and not by the architecture.
- **Large FT size causes performance drop**: Throughput drops as the FT size increases. Thus, FT size can be a bottleneck here too.
- **Smoother Curves**: The curves for the distributed setup are smoother than the single setup. It could be because the flows course through the network stack of two computers.

### B. Architectural Issues

- **Double Routing**: Packets are getting routed twice which is inefficient and TTL is decremented twice. The TTL decrementing code can lookup the identity of the incoming interface for the packet and decrement the count iff packet is coming from outside the logical router. The double routing can be avoided by adopting the architecture described in Section V.
- **Latency**: The packets in our distributed architecture travel though two software network stacks causing increased latency.
- **Routing Table Efficiency**: Forwarding table size dictates performance in certain cases. Partitioning of FT with no overlap may be desired.
- **Control Plane**: We need to ensure consistency of the router's forwarding tables inside the different computers. Else, a packet may die shuttling between the different computers of the system. A family of protocols to effect this and other control plane issues has been developed but was not implemented when our experiments were performed.
- **Inefficiency in ports per machine**: To build a router with many gigabit ports and achieve maximum performance, one PC per incoming line is necessary. Alternatively, use PCs with the newer high speed buses.
- **Fast switching interconnect**: We need to always ensure that the interconnection network is not the new bottleneck.

## V. An Alternate Distributed Architecture

Bianco et al. proposed a multistage distributed router [6] using PCs. Their design uses two stages. The first-stage acts as load balancer at layer 2 and sends incoming packets to the the back-stage. The back stage PCs act as layer 3 forwarders. In this design, there can be asymmetry between the front and back stages in terms of number of PCs, their capabilities and the interconnections.

It was observed that this architecture allows a lot more flexibility in distributing the work done by a router. The first stage may do more sophisticated processing/classification and send the packets to a particular back-end forwarder. Thus, some of the forwarders in this architecture may do specialized processing. The number of second stage PCs may be scaled as per requirements. The architecture allows redundancy and high availablilty to be implemented more easily. It must be noted that the front end stage can easily become a bottleneck in this architecture if more than a Gigabit of traffic is to flow through a first-stage PC.

A multistage design introduces some issues that need to be addressed like maintaining packet ordering [23], handling fragmented packets, synchronization of forwarding tables in the back stages, different processor capabilities in the back stages, intelligent load balancing in case of failures. Once these issues are addressed, this architecture may give us both flexibility and high performance.

## VI. Conclusion

We observe inadequacies in using a single general purpose computer as a router for our goal for high performance routing. We therefore examined a distributed router approach in order to make substantial gains in flexibility and the performance. The distributed nature allows us to implement a number of different research proposals for packet forwarding or techniques within.

We examined two such distributed architectures. The first design was a simple symmetric design that allowed

us gains in performance but not necessarily in flexibility. The second architecture, the multistage architecture, allows us to have gains in performance and in flexibility. It is scalable, robust and can be extended easily through software programming. It allows decoupling of functions inside a router. It can therefore accommodate disparate hardware and software and thus lends itself for expansion.

## APPENDIX

### A. Hardware Details

*1) For Single Computer Experiment:* Opteron 2.0GHz, 2GB DDR 400 RAM in dual channel mode. Four Gigabit Ethernet interfaces. An onboard NVidia, a DLink DL2000 32-bit PCI card, two Intel 82540EM 32-bit PCI cards. The single PCI bus runs at 66MHz. The Intel NIC cards are *NAPI enabled* in our tests.

The test machines are Intel Pentium 4 HT 3.0 GHz with 512 MB DDR 400 RAM, one Intel 82540EM 32-bit PCI GbE card. The single PCI bus runs at 66MHz.

The Layer 2 switch is a DLink DGS-1008D.

*2) For Multiple Computer Experiment:* For this case, we took three additional computers to act as forwarders. So, the four forwarders were: One Opteron 2.0GHz, 2GB DDR 400 RAM, NVidia and Intel NICs, one PCI bus at 66MHz. Two Dual Core CPUs 2.4GHz. 2GB DDR-II 400 RAM, Intel NICs, one PCI bus at 66MHz. One Dual Core CPU 2.13GHz, 2GB DDR-II 400 RAM, Intel NICs, one PCI bus at 66MHz.

### B. Software Details

*1) Tools: Click*: Click version 1.5.0 compatible with Linux kernel version 2.6.16-13. *TCP Replay*: A set of tools used to work with captured tcpdump "pcap" traffic files. *Perl*: Perl scripts for manipulation of text output, running two flows at the same time, populating Linux FTs, and in post processing to generate graphs.

*2) Operating System: Linux Version* : 2.6.16-13. *Tx Queue Length*: 10000 packets. *Rx Queue Length*: 30000 packets. *Rx Window Size*: 524287 bytes. *Maximum Rx Window Size*: 524287 bytes. *Tx Window Size*: 524287 bytes.

The RX queue is shared by all interfaces and hence we kept its length at 30000 packets. Since TX queues are unique to each particular interface, we kept its size at 10000. Further increase in lengths of queues did not give us any observable benefit. The window sizes are at the maximum.

### C. Testing Procedure

*1) Method for generating single flow:* Test Machine 3 was generated packets using the Click software at a precise rate and addressed them to Test Machine 1 (TM1). A Click script at TM1 collected the packets and incremented counters for maintaining an absolute count of packets received and for calculating the average number of packets per second. After counting, the packets were dropped. To ensure that only the correct packets were being counted, we sent/received UDP packets on port 1234 only. Each single test/reading ran for approximately 10 seconds irrespective of the rate.

*2) Method for generating two flows:* Here, the earlier method was extended to utilize all four test machines. Flow 1 was between Test Machine 3 and Test Machine 1. Flow 2 was between Test Machine 4 and Test Machine 2. The rest of the details are similar to Section C.1.

*3) Method for generating randomly destined packets:* A Click script at the source machine was used to generate packets with random destination IP addresses. Another Click script was run at the receiver to get the statistics. The rest of the steps are the same as above.

## REFERENCES


[1] V.P. Kumar and T.V. Lakshman et al., "Beyond best effort: Router architectures for the differentiated services of tomorrow's internet," *IEEE Communications Magazine*, vol. 36, no. 5, pp. 152–164, 1998.

[2] M.S. Blumenthal and D.D. Clark, "Rethinking the design of the internet: the end-to-end arguments vs. the brave new world," *ACM Trans. Inter. Tech.*, vol. 1, no. 1, pp. 70–109, 2001.

[3] A. Bianco and J.M. Finochietto et al., "Open-source pc-based software routers: A viable approach to high-performance packet switching," in *Proc. of the 3rd International Workshop on QoS in Multiservice IP Networks*, 2005, pp. 353–366.

[4] L. Torvalds, "The linux operating system," http://www.kernel.org.

[5] E. Kohler and R. Morris et al., "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[6] A. Bianco and J.M. Finochietto et al., "Scalable layer-2/layer-3 multistage switching architectures for software routers," in *Globecom 2006*, 2006.

[7] D.L. Tennenhouse and J.M. Smith et al., "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80–86, 1997.

[8] S. Dharmapurikar and P. Krishnamurthy et al., "Deep packet inspection using parallel bloom filters," *IEEE Micro*, vol. 24, no. 1, pp. 52–61, 2004.

[9] S. Kumar and S. Dharmapurikar et al., "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 339–350, 2006.

[10] T. Spalink and S. Karlin et al., "Building a robust software-based router using network processors," in *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, New York, NY, USA, 2001, pp. 216–229, ACM Press.

[11] J.H. Saltzer, D.P. Reed, and D.D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems*, vol. 2, no. 4, pp. 277–288, Nov. 1984.

[12] S. Keshav, *An Engineering Approach to Computer Networking*, Pearson Education, 1997.

[13] S. Karlin and L. Peterson, "Vera: an extensible router architecture," *Computer Networks*, vol. 38, no. 3, pp. 277–293, 2002.

[14] Y. Gottlieb and L. Peterson, "A Comparative Study of Extensible Routers," in *2002 IEEE Open Architectures and Network Programming Proceedings*, NY, USA, June 2002, pp. 51–62.

[15] M. Handley, O. Hodson, and E. Kohler, "Xorp: An open platform for network research," in *Proc. of the 1st Workshop on Hot Topics on Networks*, 2002.

[16] J. Rexford and A. Greenberg et al., "Network-wide decision making: Toward a wafer-thin control plane," in *ACM SIGCOMM HotNets'04*, 2004.

[17] IETF Working Group for Forwarding and Control Element Separation, ," http://www.ietf.org/html.charters/forces-charter.html.

[18] C. Partridge et.al., "A 50-gb/s ip router," *IEEE/ACM Transactions on Networking*, 1998.

[19] A. Kumar, D. Manjunath, and J. Kuri, *Comunication Netwrking: An Analytical Approach*, Elsevier, 2004.

[20] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, "Scalable high speed IP routing lookups," in *Proceedings of SIGCOMM '97*, September 1997, pp. 25–36.

[21] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE Network*, vol. 15, no. 2, pp. 24–32, Mar/Apr 2001.

[22] J. Mogul and S. Deering, "Path mtu discovery," 1990, Internet Request For Comments.

[23] I. Keslassy and N. McKeown, "Maintaining packet order in two-stage switches," in *Proceedings of Infocomm*, 2002, pp. 1032–1041.