IPv4 and IPv6 Forwarding Performance Evaluation on Different Operating Systems

Eric Gamess and Karima Velásquez

Universidad Central de Venezuela
Facultad de Ciencias, Escuela de Computación,
Los Chaguaramos, Caracas, Venezuela.
egamess@kuaimare.ciens.ucv.ve, karima.velasquez@ciens.ucv.ve

Abstract. PC-based routers are becoming an alternative to COTS routers, since they are cheaper and more flexible. Many times, in COTS routers, it is difficult to have access to details about internal operations and to perform any kind of interventions more complex than configuration of parameters. It is possible to transform a regular PC with several network adapters, and a widely used operating system, into a powerful router where users can install many kinds of tools for monitoring and tuning the network. Hence, due to the availability of a variety of PC operating systems, it is important to evaluate the forwarding performance for both, IPv4 and IPv6 stacks. In this paper, we analyze the performance of the IP forwarding engine of different operating systems and report results such as latency, throughput, and packet loss rate in a representative set of scenarios based on RFC 2544 specifications.

Keywords: IPv4, IPv6, Benchmarks, Performance Evaluation, Packet Forwarding Engine.

1 Introduction

Routers based on commodity x86 hardware are increasingly being used for cost and flexibility reasons. That is, it is possible to build a powerful router with a cheap PC, several NICs (Network Interface Cards), and a commonly used operating system. Similarly to a COTS (Commercial Off The Shelf) router, the created router can run routing protocols such as RIP [13][14][15] (Routing Information Protocol), OSPF [16][17] (Open Shortest Path First), and BGP [18] (Border Gateway Protocol). Also, users will have more flexibility with the PC-based router, since they will be able to install powerful debugging, performance, and filtering tools.

Modern operating systems for PCs (Windows XP/Vista, Solaris x86, Linux, Mac OS, FreeBSD, etc) can be used as the operating system of a PC-based router. In a small network, routing tables can be filled by the administrator. In medium and large networks, a dynamic routing protocol is a better alternative. There are several open source projects for TCP/IP based routing protocols. Zebra[1] is a free routing software

[1] http://www.zebra.org

suite that provides implementations of RIPv1 [13], RIPv2 [14], RIPng [15], OSPFv2 [16], OSPFv3 [17], and BGP-4 [18]. Quagga[2] is a fork of Zebra and offers the same dynamic routing protocols than its originator. XORP [26][28] (eXtensible Open Router Platform) provides a set of routing protocol implementations, an extensible programming API, and configuration tools. The supported protocols are RIPv2 [14], RIPng [15], OSPFv2 [16], OSPFv3 [17], BGP-4 [18], PIM-SM [24] (Protocol Independent Multicast Sparse-Mode), IGMP [20][21] (Internet Group Management Protocol), and MLD [22][23] (Multicast Listener Discovery). Zebra, Quagga and XORP are generally installed in a Unix based computer that will be running as a router. Vyatta[3] is a complete, open source, Linux-based router system. It is not only an operating system based on Linux with routing protocol daemons, but it is also a firewall and a VPN (Virtual Private Network) solution.

Due to concerns over the depletion of the current pool of Internet addresses and the desire to provide additional functionality for modern devices, an upgrade of the current version of the Internet Protocol (IP), called IPv4 [9], has been standardized by the IETF (Internet Engineering Task Force). This new version, called IPv6 [1][4][5][10] (Internet Protocol version 6), resolves unanticipated IPv4 design issues (exhaustion of the IPv4 address space, autoconfiguration, security at the IP level, better support for quality of service, etc). According to the ICANN (Internet Corporation for Assigned Names and Numbers), the IPv4 address space will run out by 2011. Hence, in recent years, IPv6 has received significant attention from researchers, educational institutions, software vendors, and end users. However, only a few works had been presented to compare the performance of IPv4 and IPv6 at the operating system level or at the packet forwarding engine level.

In this paper, we perform and extensive comparative study of IPv4 and IPv6 forwarding engine for different operating systems. We chose Solaris 10, Windows XP SP2, Windows Vista SP1, and Debian 4.0r3 since they are widely used. As outlined in [19], all the forwarding functions of a Linux-based system are developed inside the Linux kernel. For this reason, we did not used open source routers such as Vyatta since they are Linux-based and must have an overall performance closed to Debian.

The paper is organized as follows: Benchmarking scenario and IP forwarding performance results are reported in Section 2. Related works are presented in Section 3. Conclusion and future work are discussed in Section 4.

2 Forwarding Performance Evaluation on a PC-based Router

For our experiments, we used two types of PCs:

- Testers: These were traffic generators and measurers. That is, PCs that generate IPv4 and IPv6 packet traffic and measure latency, throughput, packet loss rate, etc. All these PCs were identical and equipped with a dual AMD Opteron 246 (2 GHz) processor, 2 GB of RAM, a 72 GB hard disk, and a 100 Mbps PCI Ethernet adapter (Intel PRO/100 S). In each tester, we installed Debian 4.0r3.

[2] http://www.quagga.net
[3] http://www.vyatta.org

- Router (or PC-based router): We had a unique PC of this type. At the hardware level, it was identical to the testers but it had four PCI Ethernet adapters (Intel PRO/100 S). In the PC-based router, we made four partitions on the hard disk and installed Solaris 10, Windows XP SP2, Windows Vista SP1, and Debian 4.0r3. The forwarding engine was activated in this PC for all the operating systems in both stacks (IPv4 and IPv6).

To deeply analyze the forwarding performance of the different operating systems, we chose a reasonable set of test-beds and benchmarks with an increasing level of complexity. The goal of our experiments is to measure latency, throughput, and packet loss rate in those selected test-beds.

2.1 Latency introduced by the PC-based router

To compute the latency introduced by the PC-based router, we setup a test-bed consisting of two testers connected through the PC-based router as shown in Figure 1. We measured the one-way delay (half the round-trip time) of an IP packet sent between the two testers. Then, we removed the router from the test-bed and measured the one-way delay of an IP packet sent between the two testers connected in a point-to-point connection (using a crossover cable). The difference between the two measurements gives the latency introduced by the PC-based router.

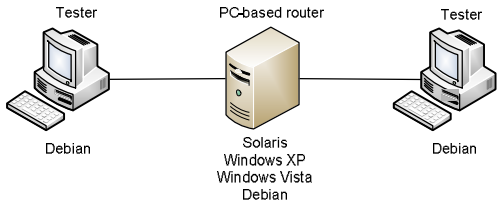

Figure 1: Initial test-bed to compute the latency introduced by the router

We did our own benchmarks using the C programming language to measure the one-way delay of a packet sent between the two testers. The benchmarks are based on the client/server model. Basically, a packet (IPv4 or IPv6) of a fixed length is exchanged between the client and the server a number of times. We took a timestamp before and after the exchange. The difference of the timestamps is divided by the number of time the packet was sent and received and by 2 to get the average one-way delay. The experiment was repeated to get consistent measurements. We compiled the source code with the gcc compiler (version 4.1.2) included in Debian 4.0r3.

As recommended in [2][3], for IPv4 packets, we chose IP payload sizes of: 8, 16, 32, 64, 128, 256, 512, 1024, and 1480 bytes. The last IPv4 payload size (1480 bytes) corresponds to the MTU (Maximum Transfer Unit) which is 1500 bytes for Ethernet. For IPv6 packets, we chose IP payload sizes of: 8, 16, 32, 64, 128, 256, 512, 1024, and 1460 bytes. The last IPv6 payload size (1460 bytes) corresponds to the MTU. Table 1 gives the results obtained for IPv4 and IPv6. We can observe that the latency

for 8 and 16 bytes are almost the same for the four operating systems in IPv4. This is due to the padding, since the minimum Ethernet payload must be 46 bytes. That is, Ethernet will randomly fill its payload up to 46 bytes for these two IPv4 payload sizes. For IPv6, Ethernet will not have to pad the payload.

Solaris and Debian have a similar latency. Windows XP and Windows Vista significantly increase their latency for an IP payload greater than or equal to 128 bytes.

Also, the latency obtained by IPv6 is superior to the one shown by IPv4 for the four operating systems. However, the difference is small and was due to the IP header (20 bytes in IPv4, and 40 bytes in IPv6).

Table 1: Latency introduced by the PC-based router in IPv4 and IPv6 (µs)

payload (bytes)	Solaris 10		Windows XP SP2		Windows Vista SP1		Debian 4.0r3	
	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6
8	32.33	33.95	29.60	31.83	37.38	37.56	31.54	31.72
16	32.48	34.02	29.84	31.85	37.47	37.60	31.91	32.03
32	32.99	34.31	30.32	32.94	38.52	38.74	32.13	32.70
64	34.45	35.86	33.18	33.29	38.91	39.29	33.71	35.61
128	39.55	41.04	347.32	348.09	352.69	353.12	39.02	41.19
256	51.45	52.76	360.27	364.02	366.85	367.40	50.92	52.74
512	73.55	74.10	381.90	383.71	387.54	388.57	74.46	75.08
1024	115.39	116.74	424.21	427.21	428.32	429.59	118.23	120.24
MTU	141.49	153.80	447.97	462.54	451.62	465.03	158.56	160.67

2.2 Throughput

Netperf[4] is a benchmark that can be used to measure the performance of many different types of networks. It provides tests for both unidirectional throughput, and end-to-end latency. The environments currently measurable by Netperf include TCP and UDP via BSD sockets for both IPv4 and IPv6. We used Netperf version 2.4.4 to estimate the unidirectional throughput between two testers connected through the PC-based router (see Figure 1). The benchmark is based on the client/server model. The client generates a simplex packet traffic at the highest possible rate and a constant payload toward the server to compute the throughput. We repeated the experiments for different IP payload as recommended in [3]. Figure 2 shows the IPv4 throughput for the four operating systems. For small values of the data sizes, Windows XP and Windows Vista have the best throughput, while Solaris has the lowest throughput. However, Solaris equals or exceeds the other three operating systems for large IPv4 payload.

[4] http://www.netperf.org

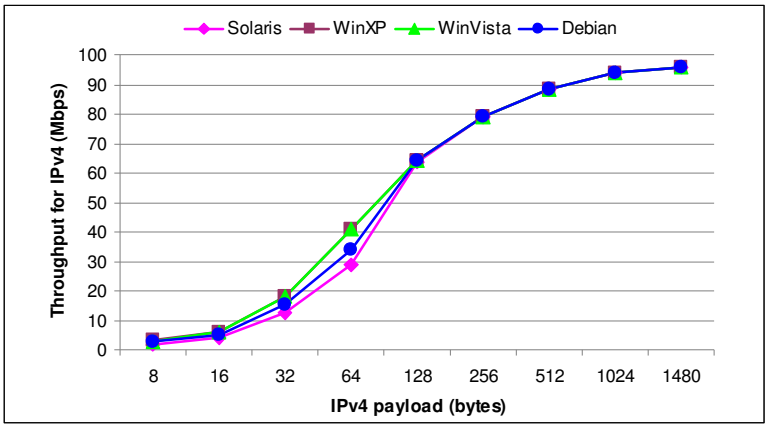

Figure 2: Throughput for IPv4 (Mbps)

A similar behavior is observed for IPv6 as shown in Figure 3. Also, the IPv4 throughput is over the IPv6 throughput for the four operating systems. However, the difference is small and is due to the IP header (20 bytes in IPv4, and 40 bytes in IPv6).

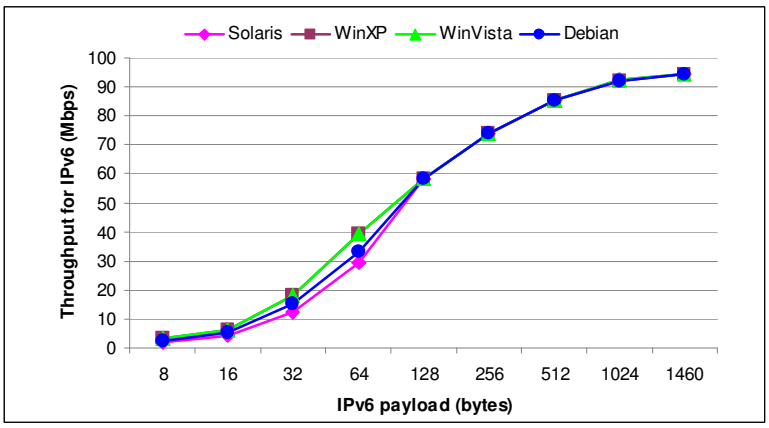

Figure 3: Throughput for IPv6 (Mbps)

2.3 Packet loss rate in a partial mesh traffic network

To evaluate packet loss rate in partial mesh traffic network, we connected four testers to the router. Three testers were acting as generator and one tester was measuring the packet loss rate (see Figure 4). We used D-ITG [25] (Distributed Internet Traffic Generator) which is based on the client/server model and allows

computing the packet loss rate. There are many ways to configure the client (ITGSend) and the server (ITGRecv) in D-ITG but we did the following for this experiment: each client generates a simplex packet traffic with a constant inter departure rate and a constant IP payload (1024 bytes) toward the server. We repeated the experiment for different constant inter departure rate as recommended in [3].

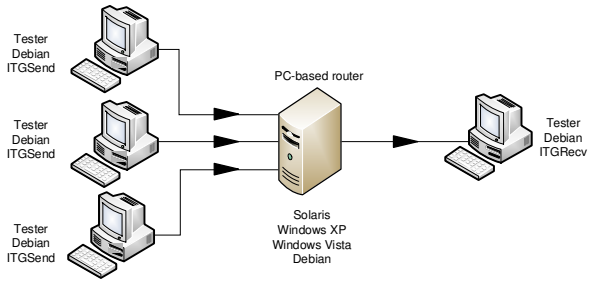

Figure 4: Test-bed for the packet loss rate in a partial mesh traffic network

For IPv4, the results of our experiments (see Figure 5) seem to indicate that Windows Vista has a better IP forwarding management system that the other three operating systems since its packet loss rate is inferior for alls the constant inter departure rate tested. Similar results were obtained for other IPv4 payload length (32, 64, 128, 256, and 512 bytes).

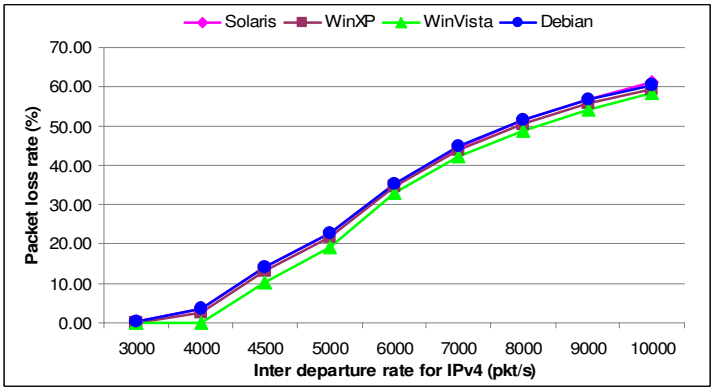

Figure 5: Packet loss rate for IPv4 in a partial mesh traffic network

Figure 6 shows the results obtained with IPv6 for a constant IPv6 payload of 1024 bytes. In this case, it seems that the IP forwarding management system has similar performance for the four operating systems. Similar results were obtained for other IPv6 payload length (32, 64, 128, 256, and 512 bytes).

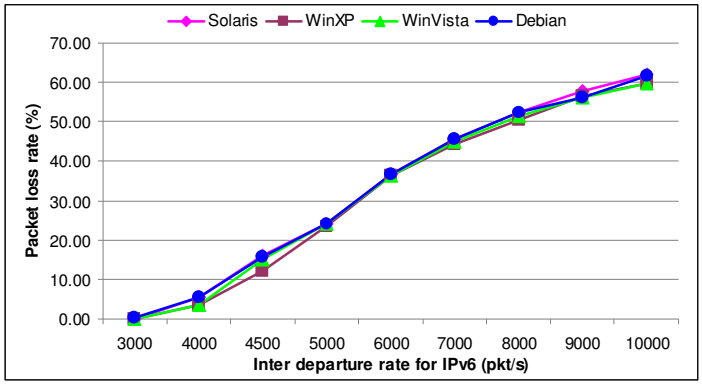

Figure 6: Packet loss rate for IPv6 in a partial mesh traffic network

2.4 Packet loss rate in a full mesh traffic network

One of the most important experiments is to stress the PC-based router under heavy network load. To evaluate this scenario, we connected four testers to the router. All the four testers were acting as generator and measurer at the same time. We used D-ITG and each tester created a simplex flow of packet at the highest possible rate and a constant payload toward the other three testers, that is, we created a full mesh traffic network. We measured the average packet loss rate for different IP payload length.

Figure 7 shows the results for IPv4. Windows Vista seems to manage the stress better than the other three operating systems since it has the lowest packet loss rate for all the IPv4 payloads tested. Also, and for all the four operating systems, it is noticeable that the PC-based router is able to manage all the traffic for large frames, but drops many packets for small frames.

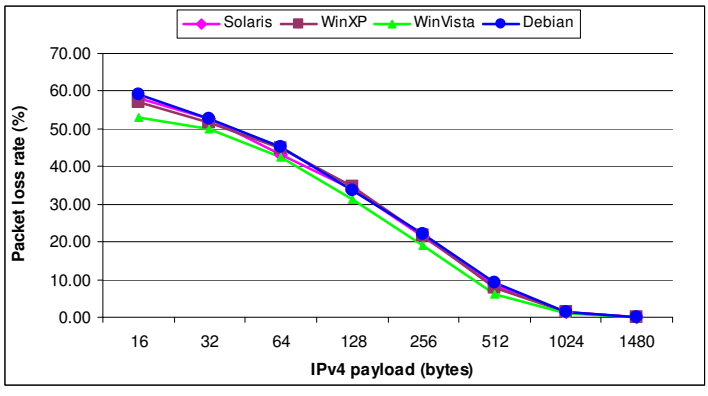

Figure 7: Packet loss rate for IPv4 in a full mesh traffic network

For IPv6 (see Figure 8), the four operating systems has a similar behavior. Also, we can observe a poor behavior of the four operating systems to manage small packets under stress.

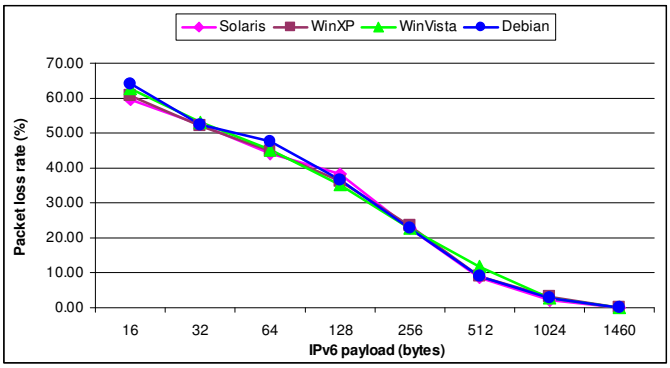

Figure 8: Packet loss rate for IPv6 in a full mesh traffic network

3 Related Works

Ettikan et al. [6][7] analyzed IPv6/IPv4 performance using simple applications (ping and FTP). They used computers with FreeBSD and KAME IPv6 protocol stack to simulate routers. They reported latency using the ping utility, and throughput using the FTP application. Zeadally et al. [11] evaluated IPv6/IPv4 performance on Windows 2000, Solaris 8, and RedHat 7.3. The authors experimentally measured throughput of TCP and UDP, latency, CPU utilization, and web-based performance characteristics. Mohamed et al. [8] evaluated IPv6/IPv4 performance on Windows 2003, FreeBSD 4.9 and RedHat 9. They measured throughput, round-trip time, socket-creation time, TCP-connection time, and number of connections per second in three different test-beds. The first test-bed consisted of a single computer and communication was limited to processes running in this computer using the loopback interface. In the second test-bed, two computers were connected through an Ethernet hub. The Ethernet hub was replaced by a router in the third test-bed. Gamess and Morales [12] setup two test-beds to evaluate IPv6/IPv4 performance stacks. In the first test-bed, they connected two identical PCs with a point-to-point link to estimate the TCP and UDP throughput of three important operating systems (Windows XP SP2, Solaris 10, and Debian 3.1) for IPv4 and IPv6. They inferred from this experiment that the IPv4 throughput is over the IPv6 throughput for both, TCP and UDP in all the three operating systems. In the second test-bed, they used five identical routers (Cisco Systems 2811) with 256 MB of RAM and two FastEthernet interfaces. They connected two identical PCs through a chain of routers (0 to 5) to study the impact of intermediate devices over the TCP and UDP throughput for both protocols (IPv4 and IPv6) in large networks where hosts can be separated by a few intermediate layer-thee devices. Bolla and Bruschi [27] reported results carried out on open router

architectures based on Linux software using different optimizations. They presented a detailed scheme of the forwarding code in Linux Kernel 2.6 versions and limited their work to IPv4. For their tests, they used a professional device[5] (Agilent N2X Router Tester), which can be used to obtain throughput and latency measurements with very high accuracy levels (guaranteed timestamp resolution of 10 ns). They also did some measures of CPU utilization in order to identify bottlenecks in the open routers.

Excepting the work of Bolla and Bruschi [27], all the previous researches are focused on the evaluation of the performance of the IPv6/IPv4 stacks at the operating system level or for end user applications. Bolla and Bruschi [27] evaluated the IP forwarding performance for open routers based on Linux, but their work is bounded to IPv4. In our contribution, we make an analysis of the forwarding performance of PC-based routers using different operating systems for both, IPv4 and IPv6.

4 Conclusion and Future Work

In this paper, we report the results of several experiments that we carried out to evaluate the IP forwarding performance of different operating systems (Solaris 10, Windows XP SP2, Windows Vista SP1, and Debian 4.0r3) running on a PC-based router. For our experiments, we used a set of RFC 2544 [3] compliant tests. Our measurements use both IPv4 and IPv6 stacks, and include latency, throughput, and packet loss rate.

We observed similar results among the different operating systems tested in our experiments. In general, the forwarding performance is better in IPv4 than in IPv6. However, the difference is small and it is due to the difference in the IP headers (20 bytes for IPv4 and 40 bytes for IPv6). Also, the PC-based router can properly handle a heavy network load for large packets, but has a high loss rate for small packets.

In some experiments, Windows Vista outperforms the other three operating systems. This is probably due to its updated implementation of the TCP/IP stack, known as the Next-Generation TCP/IP stack, which includes native implementation of IPv6, as well as complete overhaul of IPv4.

Finally, we plan to further investigate the IP forwarding performance engine of different operating systems in the presence of filtering conditions since most ACLs (Access Control Lists) are based on UDP and TCP ports which are faster to find in IPv4 packets than in IPv6 packets, due to IPv6 chained extension headers [5].

References

[1] Blanchet M. Migrating to IPv6: A Practical Guide to Implementing IPv6 in Mobile and Fixed Networks. Wiley. 1 edition. January, 2006.

[2] Bradner S. Benchmarking Terminology for Network Interconnection Devices. RFC 1242. July, 1991.

[5] http://advanced.comms.agilent.com/n2x

[3] Bradner S, McQuaid J. Benchmarking Methodology for Network Interconnect Devices. RFC 2544. March, 1999.
[4] Davies J. Understanding IPv6. Microsoft Press. 2 edition. January, 2008.
[5] Deering S, Hinden R. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460. December, 1998.
[6] Ettikan K, Gopi K, Takefumi Y. Application Performance Analysis in Transition Mechanism from IPv4 to IPv6. IWS2000. Japan. February, 2000.
[7] Ettikan K. IPv6 Dual Stack Transition Technique Performance Analysis: KAME on FreeBSD as the Case. Technical report, Faculty of Information Technology, Multimedia Univ., Jalan Multimedia. October, 2000.
[8] Mohamed S, Buhari M, Saleem H. Performance Comparison of Packet Transmission over IPv6 Network on Different Platforms. IEE Proceedings Communications 153(3): 425-433. June, 2006.
[9] Postel J. Internet Protocol. RFC 791. September, 1981.
[10] Popoviciu C, Levy-Abegnoli E, Grossetete P. Deploying IPv6 Networks. Cisco Press. 1 edition. February, 2006.
[11] Zeadally S, Wasseem R, Raicu I. Comparison of End-System IPv6 Protocol Stacks. IEE Proceedings Communications 151(3): 238-242. June, 2004.
[12] Gamess E and Morales N. Implementing IPv6 at Central University of Venezuela. In Proceedings of LANC'07 (IFIP/ACM Latin America Networking Conference 2007), San José, Costa Rica. October, 2007.
[13] Hedrick C. Routing Information Protocol. STD 34, RFC 1058. June, 1988.
[14] Malkin G. RIP Version 2. RFC 2453. November, 1998.
[15] Malkin G and Minnear R. RIPng for IPv6. RFC 2080. January, 1997.
[16] Moy J. OSPF Version 2. RFC 2328. April, 1998.
[17] Coltun R, Ferguson D, and Moy J. OSPF for IPv6. RFC 2740. December, 1999.
[18] Rekhter Y, Li T, and Hares S. A Border Gateway Protocol 4 (BGP-4). RFC 4271. January, 2006.
[19] Wehrle K, Pahlke F, Ritter H, Muller D, and Bechler M. The Linux Networking Architecture: Design and Implementation of Network Protocols in the Linux Kernel. Prentice Hall. May, 2004.
[20] Fenner W. Internet Group Management Protocol, Version 2. RFC 2236. November, 1997.
[21] Cain B, Deering S, Kouvelas I, Fenner B, and Thyagarajan A. Internet Group Management Protocol, Version 3. RFC 3376. October, 2002.
[22] Deering S, Fenner W, and Haberman B. Multicast Listener Discovery (MLD) for IPv6. RFC 2710. October, 1999.
[23] Vida R and Costa L. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810. June, 2004.
[24] Fenner B, Handley M, Holbrook H, and Kouvelas I. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601. August, 2006.
[25] Botta A, Dainotti A, and Pescapè A. Multi-protocol and Multi-platform Traffic Generation and Measurement. INFOCOM 2007 DEMO Session, Anchorage, Alaska. May, 2007.
[26] Handley M, Hodson O, and Kohler E. XORP: An Open Platform for Network Research. ACM SIGCOMM Computer Communication Review. Vol. 33, Issue 1, pp. 53-57. January, 2003.
[27] Bolla R and Bruschi R. Linux Software Router: Data Plane Optimization and Performance Evaluation. Journal of Networks, Vol. 2, No. 3. June, 2007
[28] Handley M, Kohler E, Ghosh A, Hodson O, and Radoslavov P. Designing Extensible IP Router Software. In the proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI). May, 2005.