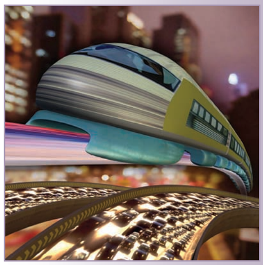

Evaluating IPV6 on Windows and Solaris

IPv6 might solve several of IPv4's shortcomings, but the longer headers and address space add overhead that affects a range of performance metrics for both TCP and UDP.

Sherali Zeadally
Wayne State University

Ioan Raicu
Purdue University

In addition to increasing the address space, IPv6 was designed to address some of IPv4's other shortcomings, including scalability and security.[1,2] The protocol also supports new Internet applications that require advanced features to provide services like real-time audio and video delivery. IPv6 is now maturing and becoming more widespread, but few researchers have empirically evaluated its performance benefits and drawbacks.

IPv6 replaces IPv4's 20-byte header with a 40-byte header that allots four times as many bits for addressing (128 bits for IPv6 versus 32 bits for IPv4). The overall impact of this increase is 1.32 percent higher overhead with IPv6 for an Ethernet maximum transmission unit (MTU) size of 1,514 bytes. Theoretically, this performance overhead is minimal, but our experimental results demonstrate that the real difference between IPv4 and IPv6 is much larger than predicted.

We established a test bed to compare the two protocol stacks along a set of six performance metrics. We also compared two IPv6 implementations running on Windows 2000 and Solaris 8 using identical hardware and settings. We performed additional tests using different configurations, including a pair of commercial routers that support dual IPv4–IPv6 stacks. While the majority of those results are beyond this article's scope, some of our experiences with the routers raised points that we address here.

Test Bed and Measurement Procedures

Most of the metrics we used in our tests characterize end-to-end application performance, providing good comparisons between IPv6-based applications and their IPv4 counterparts.

Test Bed Configuration

To focus on the protocols' behaviors on different operating systems, we connected two identical workstations using a point-to-point link, which let us eliminate most variables from the experiments (router processing, for example). Both workstations were equipped with Intel Pentium III 500-MHz processors, 256 Mbytes of SDRAM PC100, two 30-Gbyte IBM 7200 RPM IDE hard drives, and 100 Mbit-per-second (Mbps) PCI Ethernet network adapters. The workstations were set with dual-boot configurations loaded with both Windows 2000 Professional and Solaris 8.0 on two separate, identical hard drives. To add IPv6 support to Windows 2000's standard IPv4 stack, we had two add-on packages to choose from — both written by Microsoft and both in beta testing. We chose the newer release, Microsoft IPv6 Technology Preview for Windows 2000 (www.microsoft.com/windows2000/technologies/communications/ipv6/), which is supported by Winsock 2 as its programming API.

Because we were investigating the performance of IPv6 stacks at the end hosts, we initially experimented with different packet sizes. During our tests, we discovered that Microsoft's IPv6 stack for Windows 2000 could not handle fragmentation well for user datagram protocol (UDP) messages that were larger than the Ethernet MTU size of 1,514 bytes. With TCP, there was no such problem, although the stack might have other deficiencies that we did not uncover. A closer look at the problem — confirmed using the Windows 2000 Network Monitor Tool to monitor network traffic — revealed that no Ethernet frames were being delivered from the host to the wire for these large UDP messages. We could not explain why the stack was unable to handle the messages (partly because the source code was unavailable for review), but we plan to investigate this further. In contrast, Solaris 8.0 came with a dual production-level IPv4–IPv6 stack in which the IPv6 stack worked fine for UDP messages that were greater than the Ethernet MTU size.

Measurement Procedures

Because they have a direct impact on the ultimate performance perceived by end user applications, we used the following metrics in our tests:

- *throughput* (measured in Mbps) is the rate at which bulk data transfers can be transmitted from one host to another over a sufficiently long time period;
- *round-trip time* (measured in microseconds) is how long it takes a packet to travel from the sender to the receiver and back;
- *CPU utilization* (measured as a percentage) is the amount of CPU processing resources consumed by the host (sender);
- *socket-creation time* (measured in microseconds) measures how long it takes an application to create a socket;
- *TCP-connection time* (measured in microseconds) measures how long it takes an application to make a connection to a remote host application for TCP; and
- *client-server interactions* measures the number of connections that can be performed per second. For each test, the application created a socket, set up a TCP connection, sent and received a 1-byte message, and then tore down the connection and destroyed the socket.

Most tests were executed for a period of about 60 seconds, which usually generated between 50,000 and one million packets, depending on their size. We repeated each test three times using packets ranging from 64 to 1,408 bytes (typical for Internet traffic) to rule out any inconsistencies. When the results of the three tests were too inconsistent to form a valid conclusion, we performed the experiments several more times.

Experimental Results

Our results illustrate IPv6's impact on user application performance. The comparison between the Solaris and Windows IPv6 protocol stacks further shows the state of IPv6 on popular commodity operating systems.

Throughput

Throughput is valuable in understanding a system's performance. In our tests, we measured application-to-application throughput, which demonstrates the end-to-end performance that can be delivered to the end user over IPv4–IPv6 protocol stacks. As Figure 1 illustrates, Solaris shows a significant difference between IP versions' throughput for TCP messages of less than 256 bytes: IPv4 yields almost three times better performance than IPv6. The differences decrease, however, for messages of 1,024 bytes and greater. The results are quite different for Windows 2000, for which throughput is very similar for IPv4 and IPv6 with small TCP messages. For messages of more than 512 bytes, however, IPv4 yields about 11 percent higher throughput.

We obtained quite different results for UDP throughput. Figure 2 shows similar performance for

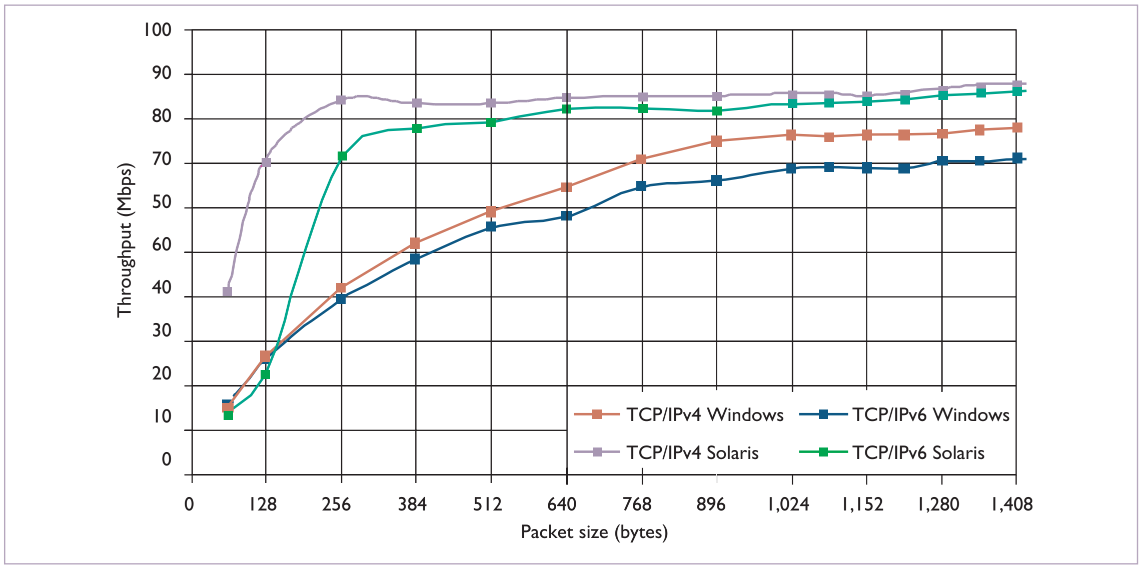

Figure 1. TCP throughput for IPv4 and IPv6 over Windows 2000 and Solaris 8.0. For small messages, IPv4 outperforms IPv6 by almost three times on Solaris.

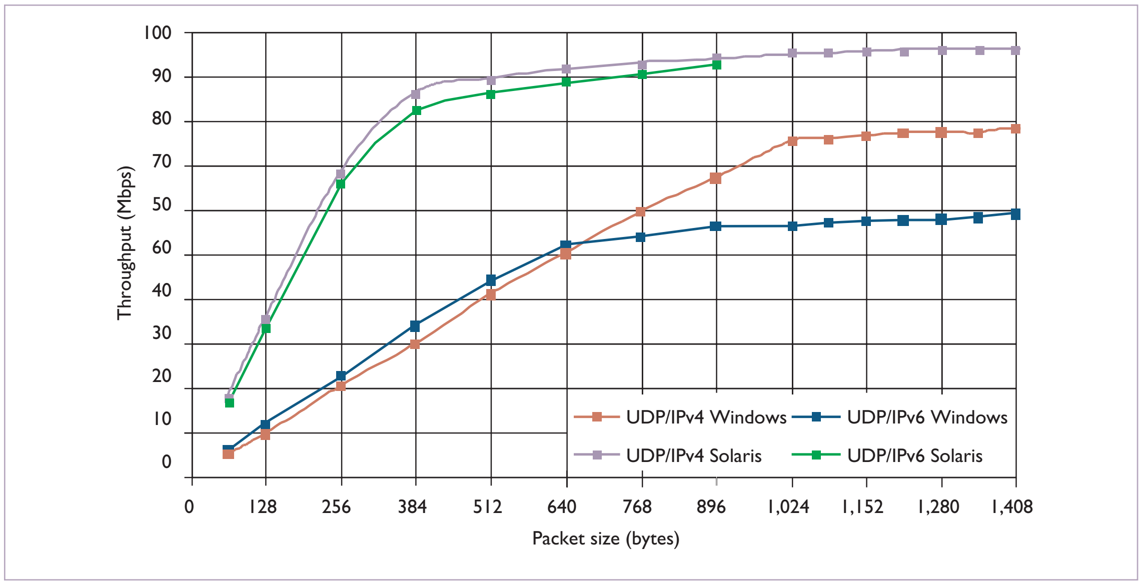

Figure 2. UDP throughput for IPv4 and IPv6 over Windows 2000 and Solaris 8.0. For small messages, IPv6 outperforms IPv4 on Windows 2000.

both IPv4 and IPv6 on Solaris — even for small messages (less than 256 bytes). Similarly, Windows 2000 exhibited similar throughput for both IPv4 and IPv6 for small messages. As message size increases, however, Windows yields lower throughput for IPv6 — 25 percent lower than IPv4 for messages of more than 1 Kbyte.

We believe that the difference in throughput performance between IPv4 and IPv6 is more pronounced for small messages with TCP than UDP mainly because of TCP optimizations such as the Nagle algorithm[6] and delayed acknowledgments. With the Nagle algorithm, the stack waits for an acknowledgment or more data from the application before sending small amounts of data with high header; with delayed acknowledgments, the stack "piggybacks" acknowledgments, which can also create sender-side delays. These optimizations affect IPv6 packet performance more than IPv4 because of the higher header overheads associated with IPv6.

Round-Trip Latency

Latency is an important performance metric for many network-based applications. Continuous media applications such as those involving audio and video are particularly sensitive to delay. Transactional applica-

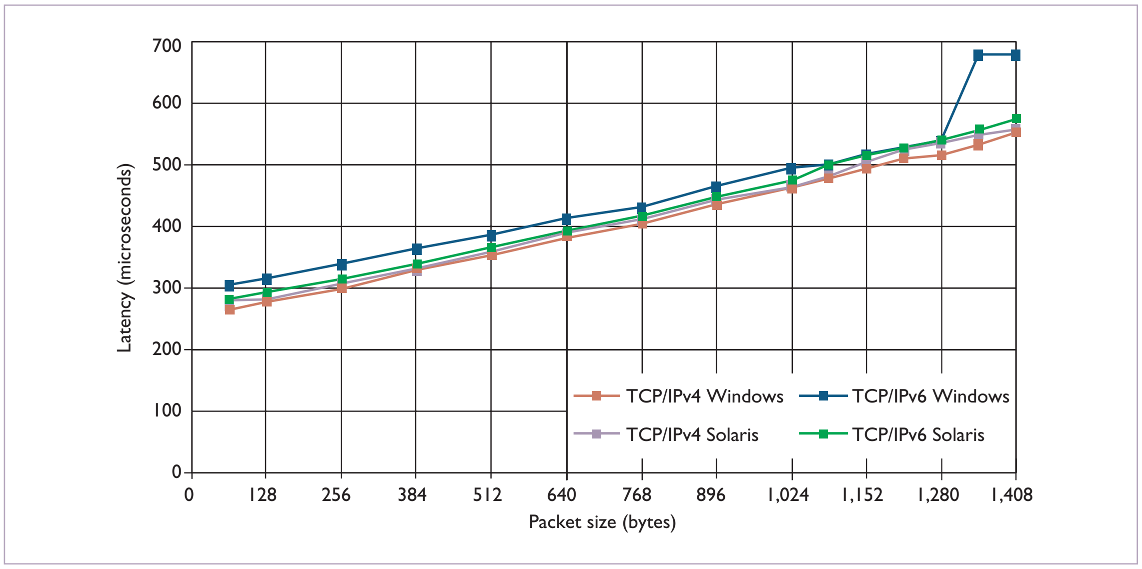

Figure 3. TCP latency for IPv4 and IPv6 over Windows 2000 and Solaris 8.0. TCP latency differences between IPv4 and IPv6 are higher on Windows 2000 than on Solaris.

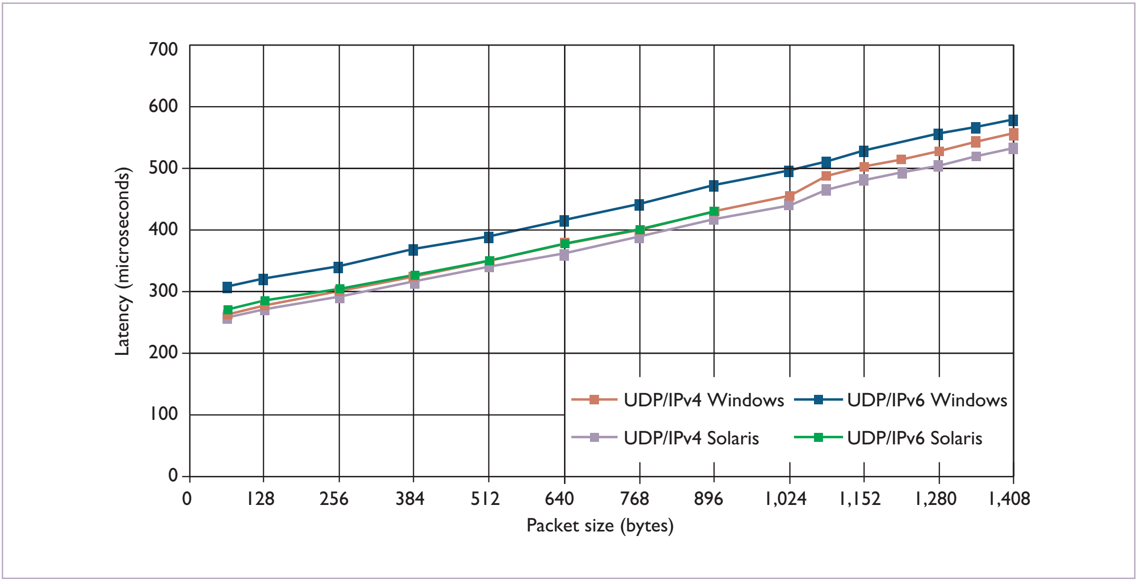

Figure 4. UDP latency for IPv4 and IPv6 over Windows 2000 and Solaris 8.0. UDP latency differences between IPv4 and IPv6 are higher on Windows 2000 than on Solaris.

tions (involving mostly request–reply operations), such as HTTP and DNS implementations, are sensitive to round-trip latency. In this context, we examine the IPv6 and IPv4 stacks' impact on end-to-end latency and application performance.

As Figure 3 shows, the TCP round-trip latency with IPv6 is about 30 percent higher (worse) than the IPv4 stack for messages up to 1 Kbyte on Windows 2000. With Solaris, we observe a 5 percent increase in latency for IPv6 compared to IPv4. For packets larger than 1 Kbyte, we see an increase in latency on both Solaris and Windows 2000 of around 1 to 2 percent using IPv6 with TCP. This is probably due to the amortization of overheads associated with larger packet sizes (larger user payloads).

The UDP results depicted in Figure 4 also show about 30 percent higher latency with IPv6 than IPv4 on Windows 2000 for messages up to 1 Kbyte. With increasing message sizes, the latency difference between IPv4 and IPv6 packets decreases. With Solaris, we again obtained about 5 percent higher latency with IPv6 than with IPv4. It is interesting to note that there were no significant latency differences between TCP and UDP (at least for the message sizes tested) for either operating

Table 1. UDP and TCP socket-creation time and TCP-connection time for IPv4 and IPv6.

Operating system	Transport protocol	IP version	Socket-creation time (microseconds)	Connection time (microseconds)
Solaris 8.0	TCP	IPv4	1,622.51	576.86
		IPv6	1,736.45	611.55
	UDP	IPv4	1,908.21	N/A
		IPv6	2,041.74	N/A
Windows 2000	TCP	IPv4	6,128.74	675.93
		IPv6	8,006.51	1,012.13
	UDP	IPv4	6,002.74	N/A
		IPv6	6,812.13	N/A

system, despite the fact that TCP has greater overhead than UDP.

CPU Utilization

CPU availability is an important resource at the end system. We measured CPU usage at the sending host during our throughput experiments, using the Windows 2000 Task Manager's performance monitor tool. We observed that for similar throughput results (nearly 90 Mbps over the 100-Mbps Ethernet local area network), TCP over IPv6 used about 20 percent more CPU resources than TCP over IPv4.

Socket-Creation Time and TCP-Connection Time

From Table 1, it is clear that Solaris 8.0 outperforms Windows 2000 for both TCP and UDP socket-creation time and TCP-connection time. Socket-creation time did not change significantly between IPv4 and IPv6 under Solaris (a 7 percent increase for both TCP and UDP); under Windows 2000, however, IPv6 socket-creation times increased by 31 percent for TCP and 13 percent for UDP. A closer look at the steps involved during socket creation helps explain the bigger difference for an IPv6 socket than for an IPv4 socket, and for Windows 2000 compared to Solaris.

On Solaris, when a user application makes the `socket()` call, control passes to the socket library, which then makes one or more calls to the underlying operating system. On Windows 2000, the Winsock 2 library (actually known as `ws2_32.dll`) provides the bridge between applications and underlying transport service providers, which implement actual transport-protocol functions. When a socket is created, the `ws2_32.dll` selects the appropriate service provider, based on the protocol description parameter; it also forwards application procedure calls involving the socket to the appropriate service provider that controls that socket.

Given that the IPv6 socket address size is constant, we hypothesize that differences in how Windows 2000 and Solaris implement sockets are primarily responsible for the large performance differences. Solaris probably has a more efficient socket layer than the `ws2_32.dll`, faster user and kernel switches during system calls, and better protocol stack implementations. These factors also likely explain Solaris's three to four times better performance for socket creation (regardless of IPv4 or IPv6). We plan to conduct more tests in the future to confirm these explanations.

TCP connection times are higher with IPv6 than IPv4 for both Solaris and Windows 2000. Table 1 reveals that Solaris yields a 6 percent increase in connection time, whereas Windows 2000 shows a 50 percent increase. The increased connection time with IPv6 is again likely due to the overhead from the increased header and address sizes. UDP does not use a connection mechanism like TCP, so we did not measure the connection time.

Web Client-Server Simulation

Over the past few years, we have witnessed the proliferation of Web servers that handle numerous transactions per second. These transactions are typically of short duration and involve operations that are mostly request–reply in nature. Each of these operations basically requires a connection set up, data transfer, and a closing of the connection. The performance delivered to Web clients therefore depends in part on how fast servers can execute these operations. Given that many Web servers will soon support dual stacks, we were interested in the performance penalty, if any, that IPv6 clients would incur compared to IPv4 clients.

Related Work in IPv6 Performance

The main motivation behind our work was the lack of performance comparisons between IPv4 and IPv6 protocol stacks at the end system. Unlike previous efforts, we performed an empirical evaluation covering the full range of basic performance metrics and both major transport protocols for two different IPv6 protocol stack implementations.

Draves and colleagues made the first attempt at developing an IPv6 protocol stack for Windows NT, but they evaluated performance only for throughput.[1] In other efforts, Xie evaluated the beta version of Microsoft Research's IPv6 protocol stack for Windows NT 4.0.[2] He measured performance by analyzing network latency, throughput, and processing overheads on a test bed comprising two Pentium machines with 100-Mbps fast Ethernet connected via an unloaded switch. However, he did not compare the Windows NT implementation with other IPv6 implementations, other operating systems, or IPv4.

Ariga and colleagues evaluated data-transmission performance over IPv4 and IPv6 using various security protocols.[3] They used end hosts with the FreeBSD 2.2.8 operating system and KAME IPv6 protocol stack, but they did not perform detailed testing based on the broad set of metrics we employ.

Ettikan and colleagues compared IPv6 to IPv4 using the ping utility (to find latency) and an FTP application (to find throughput) with the dual-stack implementation of KAME over FreeBSD.[4,5] They did not experiment with parameters such as packet size, connection time, or protocol type. (They could not perform UDP tests due to the nature of FTP.)

References

1. R.P. Draves et al., "Implementing IPv6 for Windows NT," *Proc. 2nd Usenix Windows NT Symposium*, Usenix Association, Aug. 1998.
2. P.P. Xie, *Network Protocol Performance Evaluation of IPv6 for Windows NT*, master's thesis, Calif. Polytechnic State Univ., San Luis Obispo, June 1999.
3. S. Ariga et al., "Performance Evaluation of Data Transmission Using IPsec over IPv6 Networks," *Proc. INET 2000*, Internet Soc. Press, 2000
4. K.K. Ettikan et al., "Application Performance Analysis in Transition Mechanism from IPv4 to IPv6," tech. report, Research and Business Development Dept., Multimedia Univ., Jalan Multimedia, June 2001; www.my.apan.net/ipv6/Papers/ettikan.PDF.
5. K.K. Ettikan, "IPv6 Dual Stack Transition Technique Performance Analysis: KAME on FreeBSD as the Case," tech. report, Faculty of Information Technology, Multimedia Univ., Jalan Multimedia, Oct. 2000; www.my.apan.net/ipv6/Papers/M2USIC_Perf.PDF.
6. J. Nagle, "Congestion Control in IP/TCP Internetworks," IETF RFC 896, Jan. 1984; www.ietf.org/rfc/rfc896.txt.

In our testing, we obtained 430 connections per second for IPv4 and 404 connections per second for IPv6 on Solaris 8.0, a 6 percent drop. On Windows 2000, we obtained 147 and 115 connections per second for IPv4 and IPv6, respectively, which represents a 22 percent drop. The decrease in the number of connections with IPv6 is mainly due to the rather large increase in socket-creation and connection time. It is interesting to note, however, that Solaris significantly outperforms Windows 2000 — by almost four times — and demonstrates its potential superiority for Web servers supporting dual IPv4–IPv6 stacks.

IPv6 Router Performance

Although space constraints limit this article's scope to the performance comparison we obtained using a point-to-point configuration, we have also repeated all the experiments using other network configurations involving two commercial routers between the sender and receiver hosts. Both routers — the Ericsson AXI 462 router and the IBM 2216 Nways Multiaccess Connector Model 400 — support a dual IPv4–IPv6 stack. The use of hardware routers also differentiates our efforts from others, which have primarily used software-based routers[3,4] (see the sidebar on "Related Work in IPv6 Performance"). The cost of hardware-based dual-stack routers — the two routers we used totaled US$60,000, for example — prohibits most researchers from testing IPv6's performance with real routers. Another possible reason for the dearth of IPv4-versus-IPv6 performance comparisons might be that many router vendors are still implementing IPv6 support in their "fast" forwarding architectures. Until they complete these efforts, most routers are likely to treat IPv6 packets in the "slow" path.

Our experimental results with the two routers were inconsistent, as the Ericsson router far outperformed the IBM. The IBM router's TCP throughput performance, for example, was about 30 percent worse than the Ericsson router's for message sizes similar to those used in this work (that is, up to 1,408 bytes). TCP–IPv6 round-trip latency was also 250 to 380 percent higher (worse) with the IBM router for similar packet sizes. We witnessed fairly similar performance degradation with both routers for TCP-over-IPv6 throughput and latency results on both operating systems.

We speculate that the IBM router probably included an early implementation of IPv6, which would explain its poor performance compared to the more mature Ericsson IPv6 router implementation, which was 1.5 years newer. Nonetheless, the experimental results obtained with these routers show that commercial router implementations with IPv6 forwarding code are not yet mature.

Future Work

We hope our performance results and experiences can provide insight for network, protocol, and application designers regarding the impact IPv6 protocol stacks will have on end-to-end performance in the deployment of current and future IPv6 applications. Given Linux's wide popularity and usage, we plan to conduct similar performance tests to compare its IPv6 stack with the results we've obtained so far.

Other areas we want to investigate include IPv4-to-IPv6 transition mechanisms,[5] which constitute a necessary step toward IPv6's full deployment, and various encapsulation methods used by IPv4 and IPv6 networks. Finally, we intend to exploit IPv6 features (such as the `flow label` field in the header) to investigate end-to-end QoS support over IP-based networks.

Acknowledgments

This work was supported by a grant from Sun Microsystems (EDUD-7824-000145-US). We are grateful to Ericsson for donating an AXI 462 router and to IBM for donating a 2216 Nways Multiaccess Connector Model 400 router for use in our test bed. We also thank Narendra Goel for his suggestions.

References

1. S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," Internet Eng. Task Force RFC 1883, Dec. 1995; www.ietf.org/rfc/rfc1883.txt.
2. C. Huitema, *IPv6: The New Internet Protocol*, 2nd ed., Prentice Hall, 1997.
3. K.K. Ettikan, *IPv6 Dual Stack Transition Technique Performance Analysis: KAME on FreeBSD as the Case*, tech. report, Faculty of Information Technology, Multimedia Univ., Jalan Multimedia, Oct. 2000; www.my.apan.net/ipv6/Papers/M2USIC_Perf.PDF.
4. P.P. Xie, *Network Protocol Performance Evaluation of IPv6 for Windows NT*, master's thesis, California Polytechnic State Univ., San Luis Obispo, June 1999.
5. R. Gilligan and E. Nordmark. "Transition Mechanisms for IPv6 Hosts and Routers," Internet Eng. Task Force RFC 1933, Apr. 1996; www.ietf.org/rfc/rfc1933.txt.

Sherali Zeadally is an assistant professor in the Department of Computer Science, and director of the High-Speed Networking Laboratory, at Wayne State University. His research interests include high-speed networks, wireless networks, QoS, operating system internals, and performance analysis of networks and systems. He received a BA in computer science from Cambridge University and a PhD in computer science from Buckingham University, England. Zeadally is a member of the British Computer Society. Contact him at zeadally@cs.wayne.edu.

Ioan Raicu is a PhD student in the Department of Computer Science at Purdue University. His research interests include network processors, P2P networks, IPv6 networks, QoS over IP, and wireless sensor network architectures. He received a BS and an MS in computer science from Wayne State University. Raicu is an active member of the IEEE and the International Society for Computers and their Applications (ISCA). Contact him at iraicu@cs.purdue.edu.