

Parallel Adaptive Mobile Web Clipping

Alexander Vrenios

Abstract – Wireless mobile access to the Internet is an indispensable tool for private and corporate users alike; but both industry and academia recognize the need for better service. Improvements to commercial clipping software are biased toward the needs of the service provider in the form of better billing, dynamic provisioning, etc. Improvements to academic clipping servers focus on exploiting inherent parallelism on a variety of levels but seem to have missed one that our early investigations uncovered. The syntax of the markup language that describes a typical Web page has a tree-like structure, which suggests inherent parallelism, provided the data in one branch of the syntax tree can be processed independently of the others. This research examines the potential for improved quality of service to a handheld device on a wireless link by using a parallel cluster computing architecture as an in-stream proxy server, exploiting parallelism during the markup language transcoding process.

Index Terms – Mobile Internet, Web Clipping, Cluster Server.

1 Introduction

The Internet and the World Wide Web have become popular and important assets over the last decade. Access to electronic messaging and Web site content is now an indispensable tool for businesses and individuals alike. There is an even greater potential benefit for mobile Internet access via handheld cellular communications devices, if only some seemingly insurmountable technical difficulties could be overcome. Users conditioned to accessing the Web from powerful desktop computers, over high-speed Internet connections, often expect to view a tiny Web page on their cellular telephones or their personal digital assistants (PDAs). This level of service is not possible with today's technology, nor is it likely to become available soon according to [1], [2] and [3]. How can mobile Internet users access Web content from a handheld device and get adequate service?

The typical battery-powered handheld device connects to the Internet over a voice-grade RF communications link, which is noisy and subject to dropouts and reconnections. Handheld devices, also called thin clients, offer a tiny display, a slow-speed processor, and a small physical memory. Their

monochrome screens, supporting a few short lines of text, are gradually being replaced by full-color, icon-based systems, but they do not come close to the stunning graphical quality offered by today's typical desktop browser. Web content must be transcoded, or *clipped*, for presentation on a handheld device.

Client-side clipping demands too much from an already overburdened handheld device. Server-side clipping either overburdens the Web server, or demands an extraordinary effort on the part of a Web site administrator. A generalized in-line clipping server, one located between the client and the Web server, satisfies the need for reduced content without impacting the mobile device or the Web server, but any conversion inefficiencies can add delays and can negatively affect the system's performance through increased response times. Any systemic solution must address and attempt to balance the effects of all of these issues.

Generalized in-line clipping reduces the content of an ordinary Web page into a format compatible with simple handheld devices, offering better quality of service in the form of faster access speeds and improved reliability [4]. The in-line clipping service offers the most long-term potential for mobile wireless Internet access because it reduces the impact on both the handheld device and the Web server. We propose a novel distributed computing paradigm for an in-line clipping server that adaptively balances the load among identical copies of the clipping service process components in an attempt to maximize overall system throughput,

• *A. Vrenios is a Ph.D. candidate with the Department of Computer Science and Engineering at Arizona State University, Tempe, AZ 85287. Email: alexander.vrenios@asu.edu.*

while minimizing its own impact on each mobile user's response time.

2 BACKGROUND

Wireless Internet access employs a cell phone or a personal digital assistant (PDA) over a low-speed, noisy RF link to access World Wide Web content designed for powerful desktop computers on high-speed, dedicated communications lines. Analysis of these communication pathways reveals five major areas where improvement may be proposed. Figure 1 itemizes these areas as

- Client-side Solutions
- Networking Solutions
- Proprietary In-stream Solutions
- Generalized In-stream Solutions, and
- Server-side Solutions.

Academic researchers have generally shunned all but the generalized in-stream solution in an attempt to offer an unobtrusive solution. They boast compatibility with the current handheld mobile client devices, the existing and proposed cellular networks, and the billions of Web sites with ever-changing content.

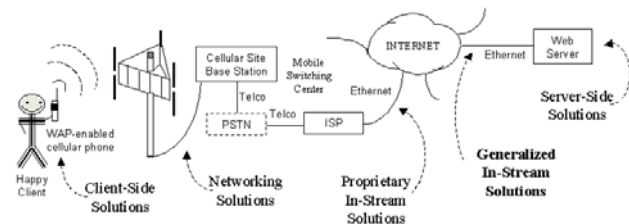


Fig. 1. Major segments along a wireless Internet access communications pathway

Generalized in-stream solutions are most often proposed as a separate computer system, a gateway, portal, proxy, or *clipping* server. (The term clipping, in this context, comes from the early information clipping services, where items of interest to a client were literally clipped from newspaper or magazine content.) Here we refer to a software process that is dedicated to reducing complex multimedia Web content into a form suitable for the tiny screens and primitive input/output mechanisms available on the typical handheld mobile access device.

Academic investigations along these lines have focused on exploiting parallelism in the process of

transcoding Web content for display on a cell phone or PDA. Web SHAKE [5] allows an ad hoc cluster of client devices to pool their individual connections into a novel multiple access protocol. It requires the incoming data stream to be multiplexed, however, by a dedicated computer that might be construed as a proxy server. BARWAN [6] proposes “islands” of scalable overlay networks that break the mold of the typical hierarchical network of local low-speed areas and interconnecting high-speed backbone networks. But they, too, have an in-stream cluster server that acts as a gateway in support of the many protocols used in wireless communication. Application-specific processes called OreOs [7] are perhaps a more direct approach to an in-stream proxy server, as are Mowgli [8], Pythia [9], Glomop [10], Digestor [11], WebTOC [12], MOWSER [13], eNetwork [14], Power Browser [15], iMobile [16], PROTEUS [17], WebTwig [18], Web Views [19], and others [20], [21], and [22]. Each of these projects attempts to exploit some form of inherent parallelism within the wireless mobile Internet access process, implements custom software, and offers direct measurement data from a test bed proxy server as evidence of its effectiveness as a solution.

We discovered one avenue of investigation that other researchers seemed to have missed. HTML is used to describe Web server content, while other markup languages (WML and HDML, e.g.) describe how to display similar data on a handheld device. If HTML and other markup languages have a tree-like syntax structure, and converting one language into another can be done on one branch independently of each of the others, then there is parallelism that can be exploited in this process.

We propose the use of a parallel computing architecture as a proxy server, and a modular parallel software architecture for the process. Such a cluster server should outperform the equivalent single-system server by performing several conversions simultaneously. The cost, of course, is the networking overhead. Each branch of an incoming HTML structure must be sent to an available worker node for independent conversion, and sent again to a joiner node for assembly into the outgoing markup language format.

3 EXPERIMENTAL APPROACH

The reality of handheld mobile Internet access devices, with their short-lived batteries, primitive input/output mechanisms, and small screens presents many challenges. The benefit of improving such a process goes far beyond the simple sports score or stock price quote, however. It wasn't that long ago when police and fire fighters had no means of personal communication; rescue and delivery workers lost access to their radio links when they left the vehicle, and foot soldiers are only now being issued individual communications devices. Mobile Internet access is about international, interpersonal communication.

The problem manifests itself at the *system* level. We propose a solution, ask a research question, and define and build a test bed where custom software will allow us to measure its costs and its benefits.

3.1 Problem Definition

The promise of mobile Internet access is being thwarted by its poor performance. Low-speed communication links, unpredictable response times, and the small screens on slow, unfriendly devices are driving customers away. Given the size of today's handheld devices, the expected increase in wireless link data rates, and the nature of Web site content, an in-line clipping server seems to be the only viable alternative that will improve the quality of service from a user's perspective. We propose to improve this through a parallel in-stream clipping server.

3.2 Research Question

What scalable computing architecture should be used for in-line clipping to provide a high quality of service to mobile wireless Internet users, without imposing unacceptable response times?

3.3 Test Bed Hardware Architecture

Figure 2 shows the clipping server in relation to support systems on the network. We employ Apache as a local Web server [23]. An external performance monitor gives the researcher at-a-glance information about network resource utilization levels, and a suite of client software processes exercise the system in a

single-query or a query stream mode. We also have the ability to control both the statistical distribution and the inter-arrival rate of the query stream [24].

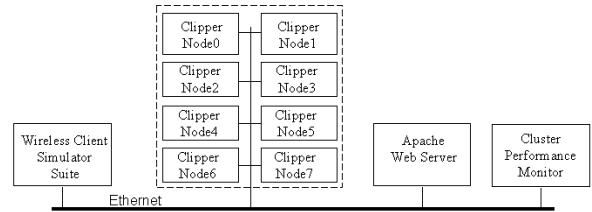


Fig. 2. Our test bed architecture showing the clipping server cluster in relation to external systems.

3.4 Test Bed Software Architecture

Clipping server software is designed with the parallel hardware system in mind. If an incoming HTML packet arrives, gets disassembled into its component parts for conversion, then reassembled before passing the result along to a handheld device, perhaps the structure shown in Figure 3 is perhaps an appropriate construct.

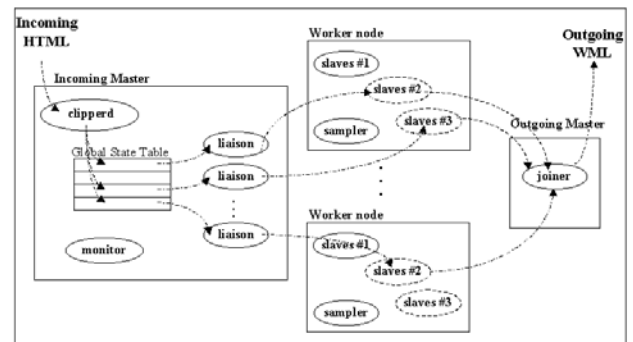


Fig. 3. Our test bed software shows the incoming query handler, several worker processes and a joiner.

Note that our software allows for multiple *slave* processes on each worker node. The number of these processes on a node is adjusted during performance measurements in order to determine the benefit from an adaptive system, which would change the number dynamically in response to varying loads. Resource utilization data are collected at each node by the *sampler* processes and sent to the *monitor* process, which keeps a summary in a local table. The main process selects a free slave on the least-busy worker node based on statistics in the table.

4 PERFORMANCE TUNING

One cannot assume that newly written software is optimal. We employed a method described in [25] that allowed us to define execution phases associated

with the processing of each incoming query packet, as shown in Figure 4.

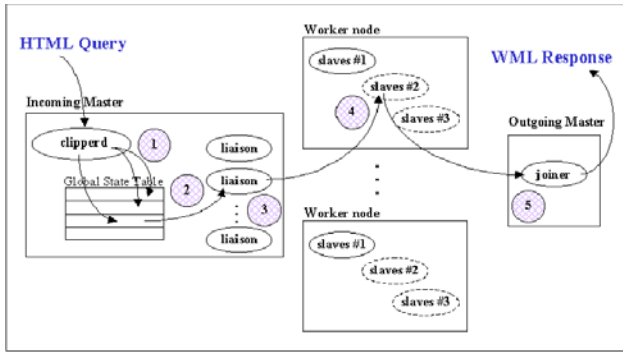


Fig. 4. The execution phases associated with the clipping service process.

We timed each execution phase and presented their average execution times as a bar graph, as shown in Figure 5.

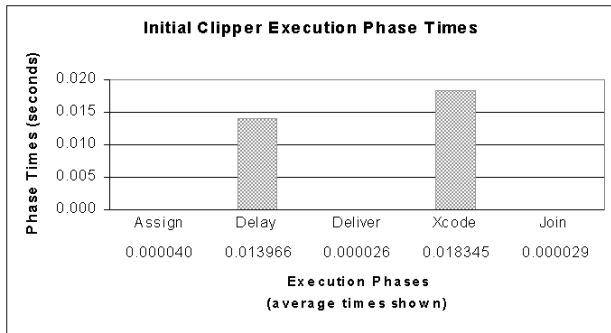


Fig. 5. Initial average execution times for each execution phase.

Investigating the source code implementation resulted in several changes that improved the overall performance of the cluster, and offered a significant improvement in phase 5, the transcoding operation. Final average execution times are shown in Figure 6.

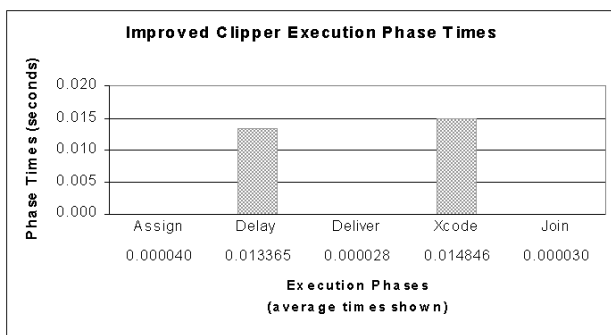


Fig. 6. Final average execution times for each execution phase.

Transcoding processing times were reduced nearly 20% through this timing and tuning process.

5 Experimental results

The performance of a cluster architecture can be plotted against the number of active worker nodes to get a sense of its scalability, but not as an indication of its advantages over the performance of a single system. The overhead associated with the resource management responsibilities of software designed to execute in a distributed systems environment are not necessary in a single system, however. Any single system comparisons must be done against a version of the software that is specially designed for a single system environment. We developed a methodology that allowed us to write software for a single system server, while hiding the sophistications of a cluster server environment inside our MSI (Manifold Server Interface), an application programmer interface [26].

The performance of a proxy server including all of the start-up and shutdown operations required at the beginning and the end of processing is valuable. Just as valuable is its performance when processing a stream of incoming packets that contains enough data to process that the effect of this start-up and shutdown overhead is minimized. We measured the performance of the single query, the query stream, and a stress test analysis of each configuration's breaking point.

5.1 Single Query Performance

Table 1 shows the average response time data we collected from a single query sent to our single system server, compared to that of our cluster server in eighteen different configurations: from one to six worker nodes, each running from one to three slave processes for transcoding.

Workers	Tasks=1	Tasks=2	Tasks=3	Distributed Optimum	Single System	Performance Improvement
1	0.597210	0.566589	0.546646	0.546646	0.582629	0.035983
2	0.555383	0.572703	0.517816	0.517816	0.582629	0.064813
3	0.569288	0.564375	0.580463	0.564375	0.582629	0.018254
4	0.575079	0.547103	0.623678	0.547103	0.582629	0.035526
5	0.624323	0.539520	0.539503	0.539503	0.582629	0.043126
6	0.546988	0.552254	0.606118	0.546988	0.582629	0.035641

Table 1. Single query response time performance.

We selected the best slaves-per-worker times in each active worker configuration and compared it to the response times from the single system. We then calculated the performance improvement and, as you can see, the cluster server outperformed the single system version in every case.

5.2 Query Stream Performance

This result required a custom version of our client software that sent a stream of packets to the servers for conversion, awaited each response, and calculated an average result for display after the run completed. Table 2 shows these response times.

Workers	Tasks=1	Tasks=2	Tasks=3	Distributed Optimum	Single System	Performance Improvement
1	0.647	0.627	0.608	0.608	0.717	0.109
2	0.702	0.613	0.680	0.613	0.717	0.104
3	0.637	0.607	0.597	0.597	0.717	0.120
4	0.646	0.720	0.671	0.646	0.717	0.071
5	0.625	0.604	0.584	0.584	0.717	0.133
6	0.696	0.613	0.662	0.613	0.717	0.104

Table 2. Query stream response time performance.

Once again, we selected the best times from each of our cluster's worker configurations and compared it to the single system's average response time. Here too, the cluster outperformed the single system version in every case.

5.4 Stress Testing

Another important metric is the breaking point of the system: the incoming query stream inter-arrival time at which packets are lost, processes hang, etc. While we expected the cluster server to survive much higher rates than the single system, we were surprised to see the single system version take the honors this time. The single system is twice as robust as the distributed server, as shown by the plot in Figure 7.

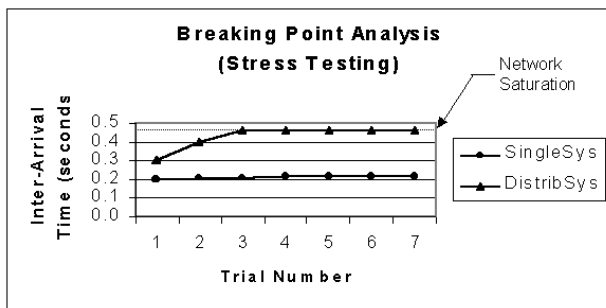


Table 3. Stress testing for breaking point analysis.

The distributed system version, you may recall, sends an extra set of packets to the slaves processes running on each worker node. After converting the data, each slave process sends its results to a joiner process for final assembly of the outgoing converted packet. Our test network exhibited a message transit time of 7 milliseconds. Our test files contained about

twenty-five branches of data to convert, resulting in fifty network messages required for every incoming packet in support of the cluster system environment.

We estimate that the network was near the 75% utilization point, which is certainly cause for failure due to saturation on a small Ethernet [27].

5.5 Scalability Analysis

Finally, as mentioned earlier, we can plot these earlier performance measure against the number of worker nodes in the cluster server's configuration. What we get (in Figure 8) is a plot of how the performance is affected by the combined virtues of added processing power, and the overhead of the added distributed system management requirement.

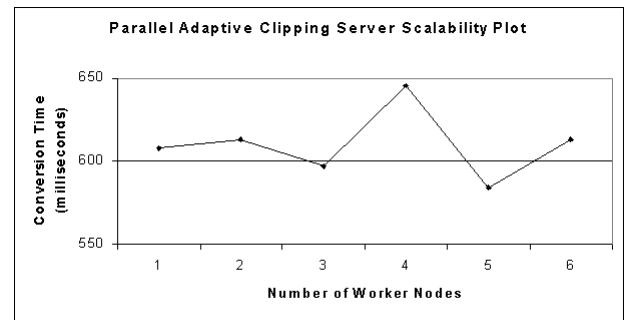


Figure 8. Scalability analysis from our performance data.

A cursory examination of this plot might prove the 5-node configuration as optimal, but the gradual descent of the final three points suggests that the response times are headed lower still, leading us to suspect that a larger cluster may reveal even better scalability than that of our test bed.

6 CONCLUSIONS

Our objective was to draw attention to the inherent parallelism in the syntactical structure of popular markup languages, like HTML, WML and HDML. If all of the advancements in computer science over the past half-century could be summed up in a single word, that word would be *overlap*. There was a time, not that long ago, when huge, expensive computers came to a halt while printing a line of characters on the printer. The exploitation of parallelism at these high levels, on down to the tiniest integrated circuits has led this technology to new heights that few could have dreamed of back then. The direction we suggested seems obvious in hindsight.

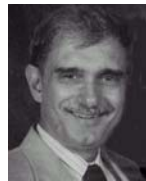
Our data show that the distributed system version of our clipping server outperformed the single system version in every case, but was less robust by a factor of two. The network was the limiting factor, causing the cluster to lose packets during processing hand-off. While the network saturation problem did not impede performance during our standard testing sequence, it is clearly an issue worth considering in any future testing plans.

Future researchers may want to extend the capabilities of their test bed to extend beyond the optimal point of scalability. They may also want to consider some form of data compression to reduce the network saturation at higher loads, or perhaps consider a much faster networking environment.

7 REFERENCES

- [1] B. N. Schilit, J. Trevor, D. M. Hilbert and T. K. Koh, "Web Interaction Using Very Small Internet Devices," *IEEE Computer*, vol. 35, no. 10, pp. 37-45, Oct. 2002.
- [2] B. N. Schilit, J. Trevor, D. M. Hilbert and T. K. Koh, "m-Links: An Infrastructure for Very Small Internet Services," in *Proc. 7th Int'l. MobiCom01*, 2001, pp. 122-131.
- [3] J. Borland. (2001, Jan.). A High-Wireless Act. *News.com*. Online. Available: <http://news.com.com/2102-1033-251073.html>
- [4] APiON, Ltd. (1999, Jun.). *Wireless Application Protocol (WAP) Tutorial*. Online. Available: <http://www.webproforum.com/wap/topic01.html>
- [5] Y. Konishi, S. Ishihara and T. Mizuno, "Web SHAKE: A Fast Access Method for Mobile Terminals on Temporary Cluster Networks," in *Proc. IEEE Conf. on Communications*, vol. 5, 2002, pp. 3439-3443.
- [6] R. H. Katz, E. A. Brewer, E. Amir, H. Balakrishnan, A. Fox, S. Gribble, T. Hodes, D. Jiang, G. T. Nguyen and V. Padmanabhan, M. Stemm, "The Bay Area Research Wireless Access Network (BARWAN)," in *Proc. IEEE COMPCON*, 1996, pp. 15-20.
- [7] C. Brooks, M. S. Mazur, S. Meeks and J. Miller, "Application-Specific Proxy Servers as HTTP Stream Transducers," in *Proc. 4th Int'l. WWW Conf.*, 1995.
- [8] M. Liljeberg, H. Helin, M. Kojo and K. Raatikainen. (1996, Apr.). Enhanced Services for World-Wide Web in Mobile WAN Environment. University of Helsinki Department of Computer Science. Helsinki, Finland. Paper. Technical Report C-1996-28.
- [9] A. Fox and E. A. Brewer, "Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation," in *Proc. 5th Int'l. WWW Conf.*, 1996.
- [10] A. Fox, S. D. Gribble, E. A. Brewer and E. Amir, "Adapting to Network and Client Variability via On-Demand Dynamic Distillation," *Operating Systems Review*, vol. 30, no. 5, pp. 160-170, Oct. 1996.
- [11] T. W. Bickmore and B. N. Schilit, "Digester: Device-Independent Access to the World Wide Web," in *Proc. 6th WWW Conf.*, 1997, pp.655-663.
- [12] D. A. Nation, C. Plaisant, G. Marchionini, and A. Komlodi, "Visualizing websites using a hierarchical table of contents browser: WebTOC," in *Proc. 3rd Conf. Human Factors and the Web*, 1997.
- [13] H. Bharadvaj, A. Joshi and S. Auelphan-wiriyakul, "An Active Transcoding Proxy to Support Mobile Web Access," in *Proc. 17th IEEE Symp. On Reliable Distributed Systems*, 1998, pp. 118-123.
- [14] B. C. Housel and D. B. Lindquist, "WebExpress: A System for Optimizing Web Browsing in a Wireless Environment," in *Proc. 2nd Int'l. Conf. On Mobile Computing and Networking*, 1996, pp. 108-116.
- [15] O. Buyukkokten, H. Garcia-Molina, A. Paepcke and T. Winograd, "Power Browser: Efficient Web Browsing for PDAs," in *Proc. Conf. on Human Factors in Computing Systems*, Apr. 2000, pp. 430-437.
- [16] C.-H. H. Rao, Y.-F. R. Chen, D.-F. Chang and M.-F. Chen, "iMobile: A Proxy-Based Platform for Mobile Services," in *Proc. 1st Workshop on Wireless Mobile Internet*, Jul. 2001, pp. 3-10.
- [17] C. R. Anderson, P. Domingos and D. S. Weld, "Personalizing Web Sites for Mobile Users," in

- Proc. 10th Int'l. WWW Conf.*, Apr. 2001, pp. 565-575.
- [18] G. Buchanan, S. Farrant, M. Jones, H. Thimbleby, G. Marsden and M. Pazzani, "Improving Mobile Internet Usability," in *Proc. 10th Int'l. WWW Conf.*, Apr. 2001, pp. 673-680.
- [19] J. Freire, B. Kumar and D. Lieuwen, "WebViews: Accessing Personalized Web Content and Services," in *Proc. 10th Int'l. WWW Conf.*, Apr. 2001, pp. 576-586.
- [20] K. Ham, S. Jung, S. Yang, H. Lee and K. Chung, "Wireless-Adaptation of WWW Content over CDMA," in *Proc. 6th IEEE Int'l. Workshop on Mobile Multimedia Communications*, 1999, pp. 368-372.
- [21] K. Kim, H. Lee and K. Chung, "A Distributed Server System for Wireless Mobile Web Service," in *Proc. 15th Int'l. Conf. on Information Networking*, 2001, pp. 749-754.
- [22] B. Knutsson, H. Lu and J. Mogul, "Architecture and pragmatics of server-directed transcoding," in *Proc. 7th Workshop on Web Content Caching and Distribution*, 2002.
- [23] G. Holden, N. Wells and M. Keller, *Apache Server Commentary*, Scottsdale, AZ: Coriolis Press, 1999.
- [24] A. Vrenios, *Linux Cluster Architecture*. Indianapolis, IN: Sams Publishing, 2002.
- [25] A. S. Tanenbaum, *Distributed Operating Systems*, Englewood Cliffs, NJ: P T R Prentice Hall, 1995, pp. 22-31.
- [26] A. Vrenios, S. Panchanathan, and F. Golshani. (2001, Dec.). Converting Software from a Single System, Concurrent Server to a Cluster Server Architecture. Presented at *Int'l Conference on Principles of Distributed Systems – OPODIS 2001*, Dec. 10-12, 2001. Manzanillo, Mexico. CD-ROM.
- [27] M. A. W. Nemzow, *The Ethernet Management Guide*, New York, NY: McGraw-Hill, 2nd ed., 1992.



Alexander Vrenios (M '89, SM '98) received his BS in Mathematics and Computer Science from University of Illinois, his MS in Computer Science from Northwestern University, and hopes to receive Ph.D. in Computer Science from Arizona State University

this May.