Personalized Email Management at Network Edges

A new technique for managing and disseminating Web-based email prefetches messages and generates dynamic pages, displaying them at the network edge. Compared to other popular Web-based email servers, the Prefetching and Caching Emails (PACE) prototype shows an improved performance with respect to user-perceived latency. Additionally, PACE's centralized neural-network-based personalized spam filter will filter spam and viruses at the server's origin, thus saving bandwidth.

Jayashree Ravi, Weisong Shi, and Cheng-Zhong Xu
Wayne State University

Almost everyone in the corporate world has at least one Web-based email account. Web-based (or HTTP-based) email systems debuted with Hotmail.com, which hosts roughly 84 million free email accounts and paved the way for Yahoo, AOL, and many others that are now largely involved in hosting such accounts. Web-based accounts have the distinct advantage of ubiquitous access — that is, users can access email at any time, from any place. However, every time they access their email, users must log on to the main HTTP server, which increases user-perceived latency in viewing email. Conventional Web caching and content delivery networks (CDNs) are of little help because email access requires authentication and every email page is personalized with users' email content; CDNs such as Akamai (www.akamai.com) pass the entire request to the origin servers and then pass the response back to the user without caching any data in between.

Another major concern for users is their email accounts being clogged with spam. Today's spam filters can be classified as *server-side* or *client-side*. Server-side filters are integrated with email servers and filter out spam at the server end. If the user has many accounts with many service providers, he or she must program the spam filter separately for each server. Client-side filters run on client machines, so users must download all their emails from the remote server machine before filters can classify messages as spam; if the number of spam messages is significant, this can use extensive network bandwidth.

Companies generally give their employees efficient server-side filters, but individuals who aren't associated with such companies must install client-side filters and pay for the time it takes to download unwanted messages. They also risk damage from viruses. Moreover, if users prefer to view their email on the Web, they face long latency times for fetching each HTML email page. Currently, no integrated spam management system exists for individuals' multiple email accounts. To solve these problems, we propose the Prefetching and Caching Emails (PACE) prototype, which prefetches email from an individual's various accounts, filters spam from all these accounts, and generates HTML pages from the network edge. It improves user-perceived latency and saves bandwidth for both the user and the service provider.

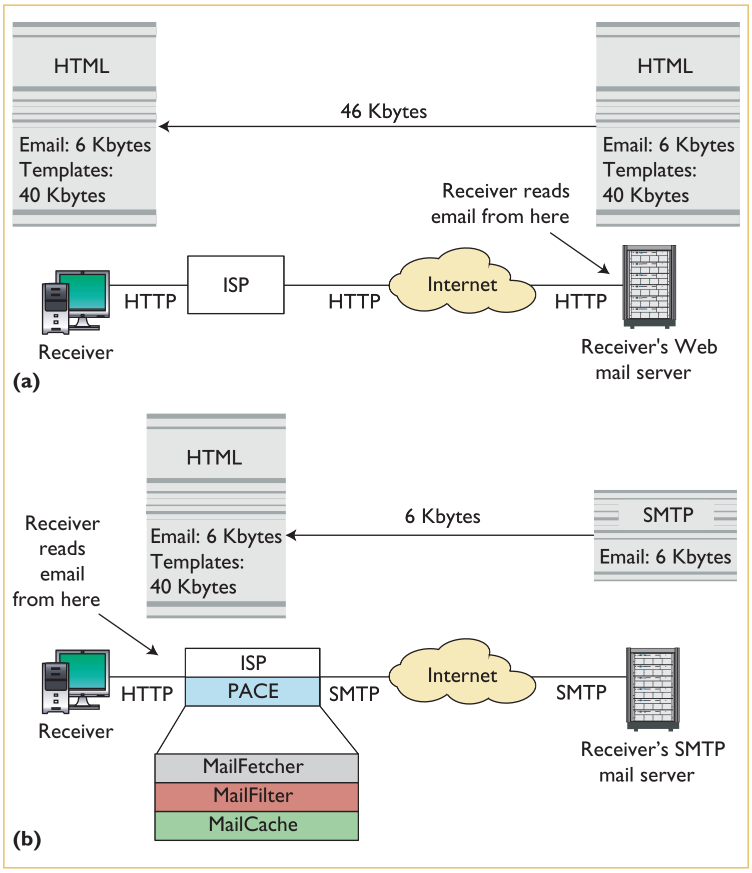

Figure 1. Typical HTTP-based email system. Flow of email for (a) a regular HTTP email system and (b) a PACE system.

Basic HTTP Email Architecture

Figure 1a shows a typical HTTP-based email system. The user logs in to the HTTP server through a secure protocol, such as HTTPS. HTTPS is secure HTTP, but it employs SSL protocol to be secure. The HTTP server is, in turn, connected to a back-end mail server that houses the user's account.

Email Page Content

We can divide a typical HTML page that contains email into three parts:

- basic email pertaining to the user;
- HTML objects such as links, text, JavaScript, and other HTML tags that the email service provider embeds with the email and that are associated with a single session key (meaning they are personalized to the user); and
- objects referred via `src` tags that retrieve images through separate HTTP `get` methods, which don't need to be associated with the user session (that is, they don't require a session key) to be retrieved.

HTML objects from popular service providers such as Yahoo and Hotmail add 18 to 40 Kbytes of data to an email, even if the email message itself is only 1 byte. We refer to these HTML objects as the *basic template*. Each time the user accesses an email, this page travels all the way from HTTP servers topologically located in distant places and passes through many networks before reaching the client (see Figure 1a). However, to access the Internet, users must first connect to an ISP. If this 18 to 40 Kbytes of data can remain at the edge of the network, which is the ISP, Internet traffic will substantially decrease, and quality of service to the user will improve in terms of perceived latency and service availability.

Whatever the email's size, the size of the provider-added template is fixed. To understand email size distribution, we collected all the email messages that our 10-member group received in their inboxes during one week. From this data, we found that more than 90 percent of the email messages our group members received were less than 6 Kbytes, and more than 80 percent of those were less than 3 Kbytes. Our analysis complies with recent analysis results using a larger group.[1] Naturally, adding the basic template to these messages increased their size considerably.

Many CDNs, such as Akamai, Digital Island, and so on, help companies disseminate their Web content to end users via caching and replication methods, which improve user-perceived latency, availability, and so forth. However, current CDN design doesn't let users manage their accounts, including personalization, preferences, and security issues (spam and virus control). The success of

Related Work in Web Caching

Our work on PACE builds on a large body of related work in the general area of Web caching, including applying dynamic content caching at edge servers, prefetching, and leveraging artificial-intelligence-based spam filters applied at edge servers.

Dynamic Content Caching

Caching dynamic pages is a challenging area in which much research is concentrated, including server-side,[1] proxy-side,[2,3] and, more recently, client-side efforts.[4] On the server side, the Data Update Propagation (DUP) algorithm maintains data dependence information between cached objects and the underlying data that affect their values in a graph.[1] When the system becomes aware of a change to this data, it applies graph-traversal algorithms to determine which cached objects are affected by the change, and then either invalidates or updates cached objects found to be highly obsolete. In this case, the Web page fragments are cached at the server.

For proxies or surrogates, researchers have proposed Edge Side Include (www.esi.org). ESI is a simple markup language used to define Web page components for dynamic assembly and delivery of Web applications at the Internet's edge. The management of cached Web page components occurs at the network edge.

Client Side Include (CSI) is a technique that allows dynamic Web page assembly from individual fragments at the client-side Web browser.[4] In this case, the Web page fragments are cached on the client machine. However, these previous works haven't proposed good solutions for caching personalized dynamic pages at the network edge. To our knowledge, ours is the first attempt to disseminate, manage, and filter personalized email at the network edge.

Prefetching

Prefetching is an efficient mechanism for reducing Web access latency,[5] which is measured by the prefetched documents' hit rate — that is, the ratio of the number of Kbytes prefetched to the number of Kbytes that the user actually views. Unlike prefetching in other scenarios, we expect our hit rate to be almost 100 percent, without any bandwidth waste, when implemented in a commercial environment because users will want to view their most recent messages daily.

Spam Filters

Among available commercial products, MailEssentials (www.gfi.com), which works on Bayesian filters,[6] has been evaluated as the best.[7] Yet, although the filter can learn the spam stream at the server level (which consists of many users), it can't be personalized for a single user. And even though Bayesian filters are excellent, artificial neural network (ANN)-based filters[8] are a more effective replacement for the human brain.[9] Surf Control (www.surfcontrol.com/products/email/), which is based on ANN, has two filters: the first recognizes and learns from the email's text pattern, and the second learns from the email's image. We find the ANN method to be very close to human spam identification. PACE has an edge-side, centralized spam filter, and because it's centrally managed, just identifying the spam in one account helps it learn and delete such email from all other accounts.

References

1. J. Challenger, A. Iyengar, and P. Dantzig, "A Scalable System for Consistently Caching Dynamic Web Data," *Proc. IEEE Conf. Computer Comm.* (INFOCOM 99), IEEE Press, 1999; www.ieee-infocom.org/1999/papers/02d_03.pdf.
2. F. Douglis, A. Haro, and M. Rabinovich, "HPP: HTML Macro-Preprocessing to Support Dynamic Document Caching," *Proc. 1st Usenix Symp. Internet Technologies and Systems* (USITS 97), Usenix Assoc., 1997, pp. 83–94.
3. W. Shi and V. Karamcheti, "CONCA: An Architecture for Consistent Nomadic Content Access," *Workshop Cache, Coherence, and Consistency* (WC3 01), 2001; www.research.rutgers.edu/~wc3/wc3-01.html.
4. M. Rabinovich et al., "Moving Edge Side Includes to the Real Edge—the Clients," *Proc. 4th Usenix Symp. Internet Technologies and Systems* (USITS 03), Usenix Assoc., 2003; www.usenix.org/events/usits03/tech/rabinovich.html.
5. C. Xu and T. Ibrahim, "Keyword-Based Semantic Prefetching in Internet News Services," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 5, 2004, pp. 601–611.
6. M. Sahami et al., "A Bayesian Approach to Filtering Junk E-mail," *Proc. AAAI-98 Workshop Learning for Text Categorization*, 1998; http://research.microsoft.com/~horvitz/junkfilter.htm.
7. M. Gunderloy, "Outrun the Avalanche," *Microsoft Certified Professional Magazine*, Oct. 2003; www.mcpmag.com/Features/article.asp?EditorialsID=362.
8. S. Haykin, ed., *Neural Networks: A Comprehensive Foundation*, 2nd ed., Prentice-Hall, 1999.
9. C. Miller, *Neural Network-Based Anti-Spam Heuristics*, Symantec, white paper, http://mn-issa.org/whitepapers/Symantec/AntiSpam%20Heuristics%20White%20Papers.pdf.

these frameworks motivated us to introduce additional edge services such as user account and spam management exclusively to cater to Web-mail service providers and provide a comprehensive common platform for merging the interests of both service providers and users.

Migrating from an existing system to PACE is simple; Hotmail already lets users download personal email through the Web-based Distributed Authoring and Versioning (WebDAV) protocol at no extra cost. The user can configure his or her Outlook Express on any client machine to synchronize with Hotmail servers. However, the client will pay for the time it takes to download email, including spam, all the way from the origin server to his or her machine. All Web-based email servers support Simple Mail Transfer Protocol (SMTP), although this service is enabled at an extra cost to the client. When we work out the logistics, we see that the service provider, the PACE host, or the client can absorb this cost. Based on who's willing to absorb it, a template suitable for the paying entity generates the page at PACE.

Spam Handling

Recent studies have shown that 40 percent of all email is spam, and the related cost in terms of money and time (for storing, disseminating, and downloading spam, and for dealing with possible virus damage) is escalating rapidly.[2]

Today's spam can't be filtered with simple filters that identify the sender's email address, black-listed mail server IP addresses, the presence of certain keywords, and so on; it has morphed considerably from messages containing straightforward spam words (obscenities, for example) that word-detecting filters could easily catch to those that hardly contain a single word a filter could identify, as every word is camouflaged with the wrong spelling or hidden between incorrect HTML tags. An effective spam filter should be able to identify a morphed spam. Humans identify spam by extrapolating acquired knowledge over a period of time: certain keywords in the subject and body are superimposed with feedback from the acquired knowledge, which helps us decide what's spam. We need a computer-based filter to replace the human decision-making process. Artificial neural network (ANN) filters come close; we've chosen an ANN-based filter for PACE and introduced two new dimensions: learning from an end user's multiple email accounts and deploying the filter at edge servers so that email messages are deleted at the origin servers in advance.

PACE's Design

As the name suggests, PACE's basic purpose is to prefetch email messages from a user's SMTP server, cache the origin mail server's templates, and dynamically generate the HTML page at the network edge.

We can demonstrate PACE's advantages with a 6-Kbyte email that a hypothetical user receives from a friend. This email is embedded in an HTML template from Yahoo that's roughly 40 Kbytes, and this page has additional object references (such as images) that make up another 120 Kbytes. Even if we assume that this 120 Kbytes, which doesn't require a session key, is cached on a nearby server, the 40 Kbytes from the HTML template isn't cached because the email and template make a single page, which is associated with a single session key and must therefore come all the way from the origin server. Thus, 46 Kbytes of data moves from the origin server all the way to the user, as Figure 1a shows. With PACE, however, only 6 Kbytes of data moves from the origin server to the ISP, and 46 Kbytes moves from the ISP to the user (see Figure 1b). The user receives the same content in both cases, but in the first case the 46 Kbytes travels all the way from the origin server to the user, passing through several Internet networks. In the second case, the user receives only 6 Kbytes from the origin server; the remainder is cached at the network edge and comes from the users' nearest Internet access point.

Because either the user or the service provider can pay for services necessary for using PACE, we consider two cases for content design: generating the page with email and a basic template or generating the page with the minimum HTML tags necessary for displaying the email alone. If the service provider is paying, it will want to add some advertisements to the email, for example, whereas the user won't want to pay for this added data.

The PACE proxy doesn't come with a mail server. Hence, users must create email accounts with their respective service providers (origin mail servers), and then enter the account information in PACE, which means our system can't replace a mail server.

In PACE, users can access email from various accounts by clicking on a separate link for each. This way, users know which email came to which account. When they reply to an email or send a new one, a separate thread is spawned that authenticates with the origin server and sends the email in the background. This is transparent to the user — that is, once the user sends the email, PACE does all the necessary work to get it to the origin mail server. It also lets users maintain common address books for use in sending email through any of their individual accounts.

Authentication

To handle user authentication, we extend the Consistent Nomadic Content Access (Conca)[3] proxy. The edge server itself can handle trust for email purposes. Users put all their login information for their various email accounts into the PACE database, and PACE uses this information to prefetch and store email. Generally, users want to check email on all their accounts at one time. Thus, a single authentication at the edge proxy will help save

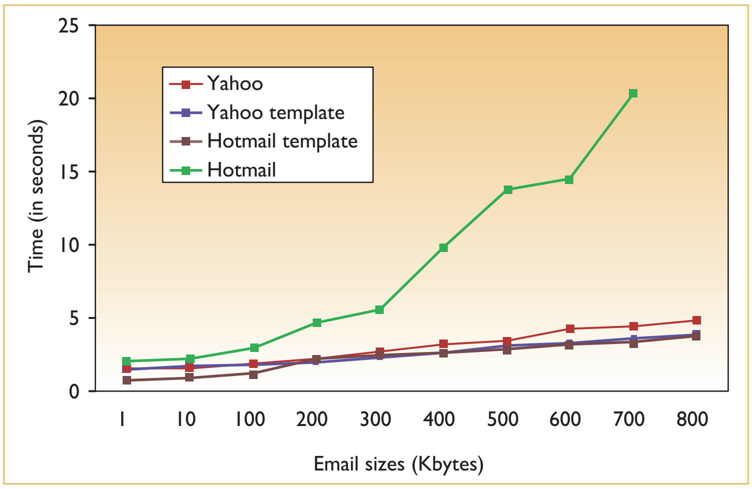

Figure 2. Latency improvement from caching templates at the network edge. The user-perceived latency improves 59 percent by caching Hotmail objects and 5 percent by caching Yahoo objects for a total email page of 100 Kbytes.

the time and bandwidth necessary to authenticate multiple accounts.

Prefetching

A prefetching program called MailFetcher runs as a daemon process, checking for new email at changeable intervals. PACE sends control signals on the order of a few bytes to check for new email; we assume that these bytes of bandwidth use will be negligible compared to the per-email savings we obtain by placing templates at the network edge. When the user logs in, a separate thread is spawned and PACE fetches all of the user's latest emails.

Spam Filtering

In addition to using ANN-based simulation to filter spam, we propose classifying a user's email as one of three types:

- *Spam* are messages a user will always delete.
- *Advertisements* are the messages that could be newsletters or promotional offers a user might be interested in during his or her spare time.
- *Email from friends and colleagues* are the messages the user will want to see as soon as they arrive.

An ANN receives a set of weighted inputs for which it computes an output that it feeds to an activation function for decisions. Given a set of inputs $x_1, x_2, x_3, ..., x_n$ with associated weights $w_1, w_2, w_3, ..., w_n$, and a threshold T_h, we can model the neuron's net output O as a function

$$O = \sum_{i=1}^{n} w_i x_i - T_h . \quad (1)$$

We've designed an ANN to receive an email in which that email's characteristics are inputs, and every characteristic is assigned an initial weight. The ANN feeds four email characteristics to the activation function. First, the ANN feeds it white list senders – that is, legitimate senders (such as friends and colleagues) that the user has identified. Second, because most spam arrives at night, the ANN feeds the activation function the time of day spam is received. Third, it feeds it the number of camouflaged and regular spam keywords detected in the subject line and email body. Spam email tries to camouflage keywords regarding pornography, get-rich-quick schemes, and so on, with incorrect spelling or HTML tags, or by hiding the keywords between HTML tags, so that filters don't catch them. Finally, the ANN feeds the activation function the number of links and images embedded in the email. The ANN readjusts the initial weight assigned to each characteristic during its self learning process in every iteration. The learning rule is defined by

$$O = \sum_{i=1}^{n} w_i x_i - T_h , \quad (2)$$

which gives the new weight on the $(k + 1)th$ iteration; d is the desired output and o is the actual output. The difference gives the error in the output for an input x_i. We define μ as the neuron's learning rate, which is set between 0 and 1.

To assist it in assigning initial weights, PACE lets users categorize any or all of the following as spam: make $$$ in your spare time, Viagra pills, pornography, mortgage discounts, and opt-out lists (someone first adds users to their list, then asks them to click on a link to opt-out). These categories have an initial list of keywords that aid the filter's first iterative step. Subsequently, the learning rules help fine tune the weights based on users' individual patterns. By studying the collected emails' characteristics, we determined that legitimate email should fall under a threshold of 0.39, advertisements should fall between 0.4 and 1.0, and spam should fall at 1.0 and higher.

Implementation and Evaluation

We designed and developed PACE at Wayne State University (WSU) during the last quarter of 2003 using open-source tools. We developed it in Java using Java server pages (JSPs; www.java.sun.com/products/jsp) and servlets for the user interface. We used a Jakarta Tomcat 3.2.3 server to generate JSP pages, and MySQL as a back-end database to store emails and user information. We evaluated our model in two environments, home and school, but due to space constraints, we present the results for the home setup only.

Experimental Setup

PACE runs on Windows XP on the server side, with a 1.6-Ghz Pentium 4 CPU and 512 Mbytes of RAM. It connects to the Internet through a Comcast Internet Cable network. The client runs Windows 98 and connects to PACE through a wireless LAN. PACE's real-world host would be ISPs connected via high-speed Internet and clusters with high-speed disk access, which would significantly improve user-perceived latency.

Evaluation of User-Perceived Latency

For our evaluation, we selected WSU's Web email service, Hotmail, Yahoo, and Rediff mail (http://in.rediff.com). The WSU mail server is on campus, the Yahoo servers are in California, and we couldn't determine the Hotmail servers' whereabouts. The Rediff mail server is in India. We expect PACE to be especially helpful in reducing cross-continental Internet traffic.

We opened separate accounts with Hotmail, Yahoo, Rediff, WSU, and PACE servers. We conducted our experiments for one week, sending two sets of email to all accounts. One set contained different-sized email messages with text-only content; the other set had different-sized emails with image-only content. (In this article, we show only the results for the text content.) We increased email sizes incrementally from 1 Kbyte to 800 Kbytes.

We designed the content for the dynamic pages PACE generates for two different customers: email service providers and users. If an email service provider used PACE, the provider would keep the template at the network edge and generate the page there. To study this, we cached Hotmail and Yahoo templates and generated the pages by fetching just the email. We compared this with the time taken to access the same email from the origin server; Figure 2 shows the results. We noted earlier that most real-world email is less than 6 Kbytes, to which the service provider adds 40 to 140 Kbytes of template and object data. Hence, considering that 100 Kbytes is the total size of most dynamic pages, we see an improvement in the user-perceived latency of 59 percent by caching Hotmail objects and 5 percent by caching Yahoo objects.

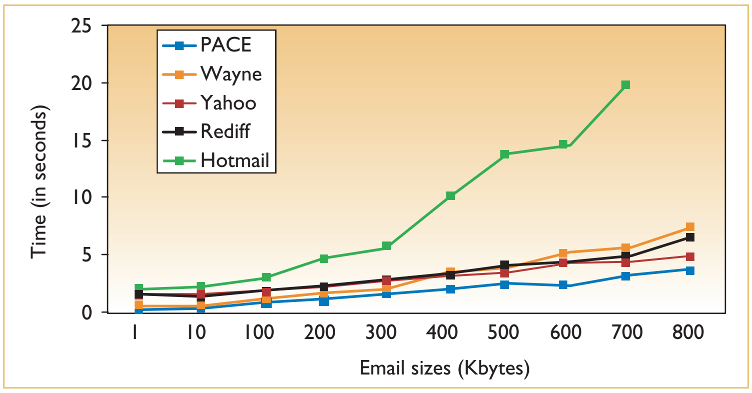

Figure 3. User-perceived latencies. Generating the basic email page without added data from the service provider improved user-perceived latency between 28 and 93 percent for a 100-Kbyte email page.

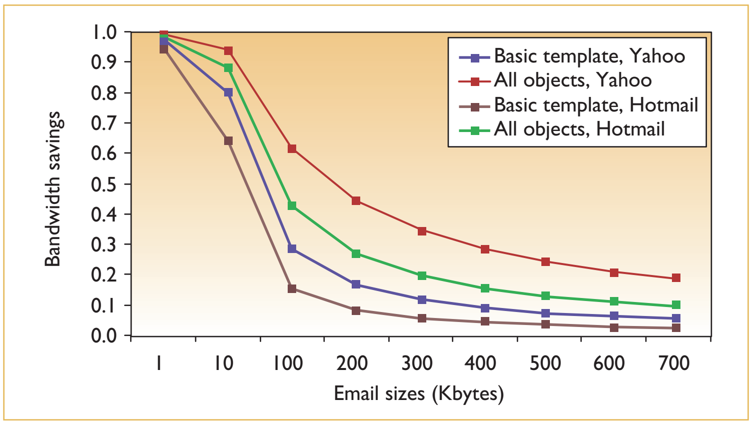

Figure 4. Bandwidth savings. Considering the popular 6-Kbyte email size, bandwidth savings from caching just the basic template is more than 86 percent in all cases. Caching all Yahoo objects gives us the best bandwidth saving.

As we described earlier, if users pay for the SMTP service, they want to see only their personal email, not advertisements that come with them. We performed user-centric evaluation by generating the page with only email. Figure 3 shows the results for this setup; PACE gave the best performance and Hotmail the worst. We see an improvement in user-perceived latency between 28 and 93 percent for a dynamic page size of 100 Kbytes.

Bandwidth Evaluation

By caching the templates and generating the dynamic page from PACE, the only bandwidth we use is for retrieving the messages alone. Figure 4 shows the bandwidth savings we achieved for different email sizes by caching the basic template and by caching the basic template and associated objects. Considering 6 Kbytes as the average email size, we achieve substantial bandwidth savings — more than 86 percent in all cases.

Spam Filtering

Neurons can be interconnected to form powerful networks that can solve complex problems. However, to highlight neural networks' power, we've implemented our model with a single neuron. We deployed our filter on our own email accounts, and filtered 90 percent of the spam and 85 percent of the advertisements. However, our aim is to implement a centralized, personal spam management at the network edge rather than evaluate spam filters per se.

We plan to integrate PACE with our Conca[3] proxy and evaluate it in a more distributed environment, which includes the ISP network and placing the proxy within the ISP by caching all types of templates and using Web services to fetch email.

We could ultimately use our proposed approach not only for personalized email accessing and delivery but also for general dynamically generated personalized Web content dissemination — for example, personalized news, stocks, and shopping Web pages — by integrating it with existing content-distribution networks. Users could enjoy the same advantages while browsing their personalized Web pages using PACE that they currently enjoy browsing static pages using CDNs today. By using PACE, we expect the access speed of a dynamic personalized Web page to be comparable to a static page's access speed.

References

1. K. Graine, "Email Archive Management 'Watching the Store,'" *The Data Administration Newsletter*, 22 Oct. 2002; www.tdan.com/i022hy03.htm.
2. J. Krim, "Spam's Cost to Business Escalates," *The Washington Post*, 13 Mar. 2003; www.washingtonpost.com/ac2/wp-dyn/A17754-2003Mar12.
3. W. Shi and V. Karamcheti, "Conca: An Architecture for Consistent Nomadic Content Access," *Proc. Workshop Cache, Coherence, and Consistency* (WC3 01), 2001; www.research.rutgers.edu/~wc3/wc3-01.html.

Jayashree Ravi is a master's student in computer science at Wayne State University. Her research interests include finding efficient techniques for dynamic Web content caching and delivery at the network edge to improve performance. Ravi received a BE in electrical engineering from the University Visvesvaraya College of Engineering, Bangalore University, India. She is a Sun Certified Java Programmer and a Microsoft Certified Professional. Contact her at jravi@wayne.edu.

Weisong Shi is an assistant professor of computer science at Wayne State University. His research focuses on dynamic Web content delivery, trusted computing, and wireless sensor networks. Shi received a BS from the Xidian University, and his PhD from the Chinese Academy of Sciences, both in computer science. He is a member of the ACM, Usenix, and the IEEE. Contact him at weisong@wayne.edu.

Cheng-Zhong Xu is an associate professor in Wayne State University's Department of Electrical and Computer Engineering. His research interests include distributed and parallel computing and scalable and secure Internet services. Xu received a BS and MS in computer science from Nanjing University and a PhD in computer science from the University of Hong Kong. He is a senior member of the IEEE and a member of the ACM. Contact him at czxu@wayne.edu.