# Understanding and Reducing Web Delays

*As the number of Web users continues to increase steadily, end-user experience has become a critical issue. The authors describe the data transfer steps involved in a typical Web session, discuss the delays associated with each of those steps, and provide some suggestions for reducing latency.*


*Mazen Zari*
*Hossein Saiedian*
University of Kansas

*Muhammad Naeem*
Sprint PCS

Research suggests that the amount of time it takes for Web pages to load is a significant factor in determining the success of a site and the satisfaction of its users. Slow performance costs e-commerce Web sites as much as $4.35 billion annually in lost revenue.[1]

Anyone who uses the Internet on a regular basis can clearly attest to the huge performance differences from site to site. A critical issue today on the Internet is perceived latency—the perceived amount of time between when a user issues a request and receives a response. In a research project, decreasing the load time of a page by approximately one second reduced the rate at which visitors abandoned a site from 30 percent to 8 percent.[1]

A study of the response times for the top ten sites in the US (derived from a Media Metrix study, http://us.mediametrix.com/data/thetop.jsp) confirmed this theory. As Table 1 shows, the number of customer hits is inversely proportional to the response time. Conventional wisdom has always suggested that faster service is better. On Web sites, small time differences can mean very different visitor retention rates. Quick response time results in higher customer hit rates.

Performance analysis and improvement research falls into two categories: work on servers and work on networks and protocols. On the server side, previous work has focused on techniques for improving server performance.[2,3] Such studies show how Web servers behave under a range of loads. These studies often suggest enhancements to application implementations and the operating systems those servers run.

On the network side, research has focused on improving network infrastructure performance for Internet applications. Studies focusing on network dynamics have resulted in several enhancements to HTTP, including data compression, persistent connections, and pipelining. These improvements are all part of HTTP 1.1.[4]

However, none of these studies look at how end-to-end system behavior interacts with network and protocol performance. There is little work on common latency sources that cause the overall delays end users experience or on techniques to help reduce their impact.

## LATENCY SOURCES

From the instant you launch a Web browser and initiate a session, a series of system and network processes take place in the background. Each process has an inherent associated latency that typically ranges from a few milliseconds (initiating a DNS lookup) to several seconds (finalizing a document transfer). Figure 1 shows a simplified model of Web communication, illustrating the request path from a client to a server.

When you enter a request for a Web site in a browser, the browser accepts the request then typically uses the Domain Name Service—a user datagram protocol service—to resolve domain names into IP addresses. The DNS process builds a connection to the DNS server to obtain the IP address. When the browser receives the IP address, it initiates the HTTP request that runs over TCP, which in turn runs over IP. The browser passes this HTTP request directly through TCP/IP, creating a socket and establishing the


0018-9162/01/$17.00 © 2001 IEEE

complete end-to-end connection. The server then fulfills the request by obtaining and serving the items that make up the page, which might include text, images, audio, and video clips.

In some situations, either the browser or the server must wait for responses from other components. The more time spent waiting, the longer the delay visitors experience while waiting for page content. The further the browser is from the server, the greater the likelihood that there will be delays. These delays can occur at any hardware or software component. Even in the best cases, each link or device in the path adds a fixed amount of time to perform its function.

## DNS lookup

DNS translates a computer name into an IP address. The purpose of DNS is to make it easier for humans to remember computer names. When a browser maps a domain name into an IP address, it sends a DNS query to a local name server. The name server may or may not have the answer in its cache. When the server doesn't have the answer, it communicates with other name servers to obtain the address.

In one study that analyzed about 13,000 Web servers, DNS lookup time remained less than 500 milliseconds for 80 percent of servers and 1 to 16 seconds for the remaining 20 percent because of timeouts and retransmits.[5] Recent advances in HTTP protocol design have reduced the number of name resolutions that the same number of HTTP requests require when using a persistent HTTP 1.1 connection.

Network managers typically operate their own DNS servers, a process that requires manually updating domain name zone data several times a day when they add domain names, upgrade servers, or conduct maintenance. According to a DNS management expert, about 75 to 80 percent of the Fortune 500 companies have some bad data in their zone that could cause configuration problems.[6] Several vendors host DNS services guaranteeing up to 100 percent availability.

## Connection

Once a domain name resolves, the client establishes a TCP connection to the server. Each router along the way does some work for every IP datagram that belongs to the particular TCP connection. First, the routing algorithm determines the IP address of the next hop. Then the router translates the IP address of the next hop into wire-level addresses. These operations might require communications with several other routers.

Each router along the path performs certain operations on every IP datagram that belongs to that particular TCP connection. These operations include consulting with the routing algorithm to determine the IP address of the next hop and translating the IP address of the next hop to a wire-level address. For example, the address resolution protocol can be used to translate the IP addresses to MAC layer addresses. These operations can also require communications with other routers. One study showed that 60 percent of the servers had less than 200 milliseconds connection time, and the remaining had connection times of 200 to 10,000 milliseconds.

**Table 1. Top 10 US Web sites.**

| # | Web site | Size (bytes) | Unique visitors | Response time (56 K) |
|---|---|---|---|---|
| 1 | AOL Time Warner network | 36,721 | 69,374 | 6.81 seconds |
| 2 | Microsoft sites | 46,846 | 59,852 | 8.62 seconds |
| 3 | Yahoo! | 27,001 | 57,522 | 9.82 seconds |
| 4 | Lycos | 15,491 | 32,384 | 6.83 seconds |
| 5 | Excite network | 47,501 | 30,535 | 12.86 seconds |
| 6 | About—The Human Internet | 55,289 | 23,588 | 13.8 seconds |
| 7 | Walt Disney Internet Group | 34,495 | 20,937 | 13.42 seconds |
| 8 | Infospace | 67,768 | 18,773 | 18.66 seconds |
| 9 | CNET networks | 77,421 | 18,435 | 22.85 seconds |
| 10 | Amazon | 86,313 | 18,046 | 25.07 seconds |

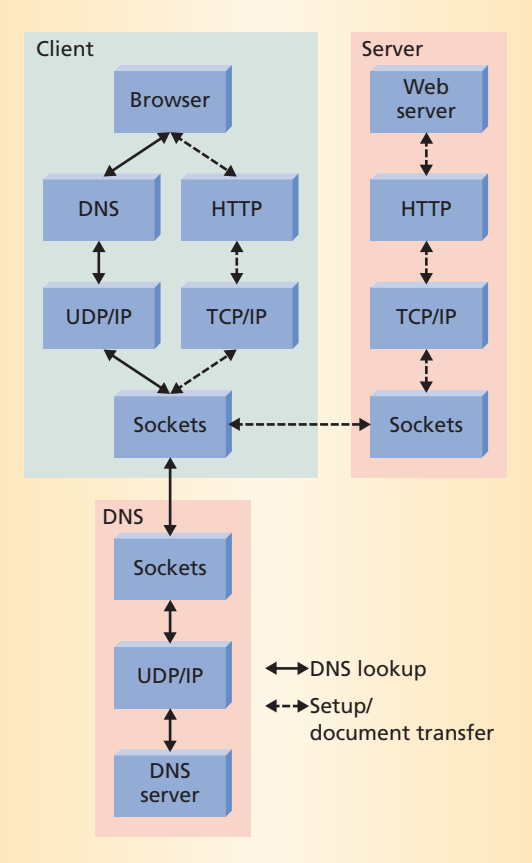


*Figure 1. A simplified model of Web communication between client and server. A typical Web transaction requires interaction between network layers and communication protocols. The client (end user) requires an additional step of domain name resolution (through a DNS lookup), which adds to the overall latency of connection setup.*

**Caching brings popular objects closer to clients so they have a shorter distance to travel and can reach the client's browser more quickly.**

### Server-side processing

A Web site services two types of information: static data, which involves a simple file fetch, and dynamic data, which involves constructing information at the server. An increasing number of sites generate dynamic content because it facilitates new services such as electronic commerce, database access, personalized presentation, and scientific computing. However, dynamic data content generation places greater I/O and CPU demands on the server. As demand for dynamic data increases, the server bottleneck becomes more critical.

In another sense, server-side processing can reduce the amount of bandwidth a Web session requires, enhancing perceived speed by sending a smaller amount of data back to the client. For example, several kinds of Web-oriented functions can be performed reasonably well using any number of technologies, such as Java, JavaScript, ColdFusion, ASP, Perl, or any other language that works in an HTML environment. However, for client-side languages such as JavaScript, the browser must download the application script to run in the user's browser. The client's browser, as a result, must deal with the application directly.

Server-side languages like ASP and ColdFusion do away with the necessity of sending application data to the client's browser. If a user requests a database entry through ASP, the only work the client's browser does is to process the HTML results that the server returns. The server actually shoulders the processing burden. Unlike Java applets, JavaScript, ActiveX controls, and several other client-side applications—all of which must move across the network before running on the client's machine—server-side languages process the application requests on the server. This means the client never has to download the application directly or indirectly.

### Document transfer

Document transfer time is the most significant contributor to delay in a transaction. Some key factors influencing document transfers are content size, bandwidth availability, proxy servers, and routing. The network between the browser and server is only as fast as the slowest link between the two. A 1-Mbyte object will always be 1,000 times slower than a 1-Kbyte object regardless of the network speed. Some ways to decrease an object's size include compression schemes, content coding, and delta encoding.

Increasing the total available bandwidth is the easiest way to decrease document transfer delays; however, increasing bandwidth has been somewhat misused, especially for overcoming congestion and delays. Furthermore, it is important to account for end-to-end bandwidth, not just bandwidth to the ISP connection. Caching servers can be deployed within a network to cache frequently used elements such as image files. Because the HTTP protocol uses a connectionless service, packets from the same document can end up taking different routes through the network. Therefore, dynamically routing packets to use the least congested network path becomes crucial.

### Experiments

We set up a test to monitor one of the top sites from Table 1. The tests monitored the total response time of the site including time spent for DNS lookup, connection time, and content download time. We conducted these tests in an environment with a 1.44-Mbps T-1 connection using a 47-Kbyte page, which downloaded in an average of 1.56 seconds. This study showed that more than 80 percent of the delay in response time can be attributed to the content download time.

Figure 2a shows that the response time varies between 0.5 seconds and 6.75 seconds, which can be attributed to varying loads on the network connections and server usage. Figure 2b shows that (excluding a few spikes) the DNS lookup time and TCP/IP connection time are not major contributors to the total response time. Figure 2c shows that content download is the biggest contributor (80 percent) to the total delay. Reducing this delay allows achieving considerable gains.

## IMPROVING PERFORMANCE

Several techniques can help improve Web performance by reducing latency. Caching—which can happen nearly anywhere, including on the server, on the user's machine, and even at ISPs and telecommunications companies—brings popular objects closer to clients so they have a shorter distance to travel and can therefore reach the client's browser more quickly. As the "Emerging Web-Caching Technologies" sidebar describes, caching is one of the most effective ways to alleviate service bottlenecks and reduce traffic over the Internet.[7]

Most popular search engines use a caching technique to quickly present information to users. Several Internet accelerator companies use DNS and content caching to offer a "faster" Web browsing experience over dialup connections.

### Caching architectures

On the ISP and telecommunications level, Web cache performance is directly proportional to the size of the client community. Increasing the size of the client community improves cache performance because it increases the probability that a cached document will soon be requested again. However, it is unrealistic to have one large cache that serves the entire community.

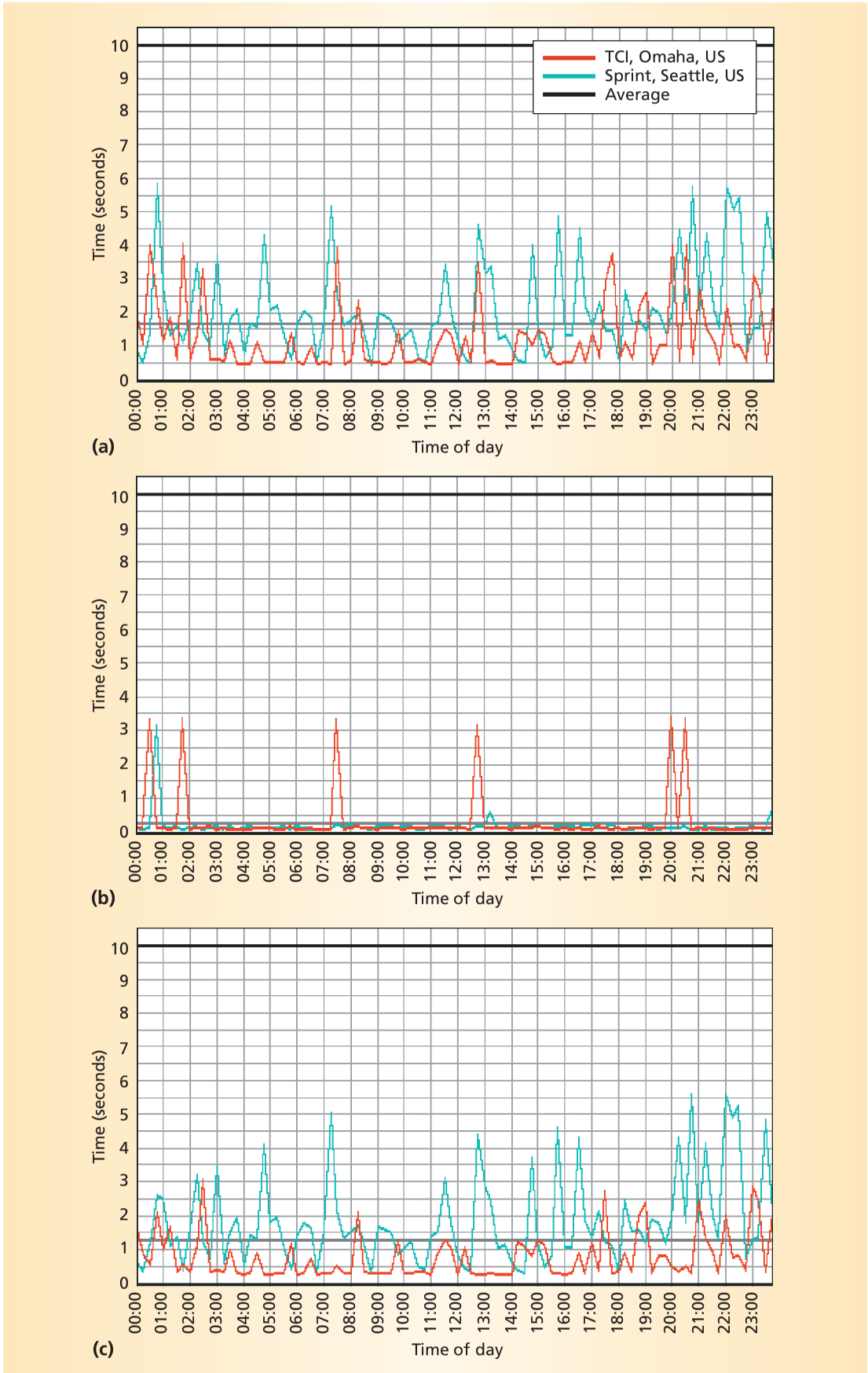


*Figure 2. Performance test on msn.com. (a) Total response time; (b) TCP/IP connection time; and (c) download time.*

## Emerging Web-Caching Technologies

*Mazen Zari and Hossein Saiedian, University of Kansas*

Web caching provides a mechanism for the temporary storage of Web objects. A Web cache monitors requests for objects between Web servers and their clients. When a client requests an object, such as an image file, the Web cache makes a local copy. When another client requests the same object, the cache makes its copy available instead of requesting another copy from the original server. In essence, Web caching reduces bandwidth consumption by reducing the overall number of requests across the network. It also reduces perceived latency by moving popular objects closer to clients. Finally, Web caching reduces server load because servers handle fewer requests.

### Multiple techniques

The Web cache performance is directly proportional to the number of clients using it. Thus, having one Web caching server handle an entire user community is unrealistic and perhaps impossible. For a large user community, several different caching techniques—including hierarchical, distributed, and hybrid techniques—have been proposed. HTTP protocols 1.0 and 1.1 provide several rules the Web cache uses to determine when to serve an object from the cache and when to make a request from the server. Additional rules, such as defining a browser's cache parameters, may be set by the cache administrator or individual users.

Many leading network companies are rushing to improve their caching technology, which depends on hit rates and algorithms that can improve overall Web site performance. Leading vendors such as Akamai, CacheFlow, Cisco, and Inktomi offer Web caching both as a product and a service.

### Static versus dynamic

Static caching only caches static components of Web objects such as images, HTML files, and text. Unfortunately, not all Web components are static, which means static caching addresses only part of the problem. Further, while static caching addresses network latency it ignores server latency. Thus, static caching should not be used as the primary way to accelerate network performance.

Dynamic caching provides a viable alternative, generating Web pages on demand, with their contents tailored to each user. Dynamic page generation typically accounts for 40 percent of the time required to deliver a Web page, however.[1] Thus, while dynamic sites provide extraordinary user experiences, the cost of using dynamic caching is so great the technique only sees use when no other method will satisfy customer needs.

Many of the Web's dynamic components are reusable, a characteristic that dynamic caching takes advantage of. According to Greg Govatos, a dynamic content accelerator caches individual page components for faster access. In this case, we define a component as a group of data that display on the page together, such as top news stories. The application server makes a request to the dynamic content accelerator for each component on a page. When the data becomes available on the accelerator's RAM-based cache, it is immediately returned to the application server in ready-to-display HTML format, thus bypassing the processing and I/O tasks normally required to create the component.[2] Govatos maintains that the efficiencies of using dynamic content accelerators stem from their lack of

- script routines,
- data-retrieval elements from local or remote databases, and
- format conversions from, for example, XML to HTML.

Network companies realize that static caching is insufficient to address user needs. Thus, much work has been done to integrate services that may include any combination of the following: static and dynamic caching, SSL-offloading, and load balancing.

Web caching remains an important and challenging area for further research. Significant challenges include cachability and identifying uncacheable objects. Some analysts estimate that as many as half of all Web objects may be uncacheable[2]—a figure that would cause the overall effectiveness of caching to suffer dramatically. We thus need to identify uncacheable objects and provide alternatives for handling them most effectively.

#### References

1. G. Govatos, "Speed up Your Dynamic Web Content," *Network World Fusion News*, 18 Dec. 2000 (www.nwfusion.com/news/tech/2000/1218tech.html, current Nov. 2001).
2. A. Wolman et al., "On the Scale and Performance of Cooperative Web Proxy Caching," *Proc. 17th ACM Symp. Operating Systems Principles*, ACM Press, New York, 1999, pp. 16-31.

A hierarchical caching architecture has multiple cache levels: bottom, institutional, regional, and national.[8] The bottom level includes the user's browser or client cache. When the browser or system does not find the user's request in the cache, it directs the request sequentially to the next higher level. When the browser eventually finds the document, the document travels down the hierarchy, leaving a copy at each cache level along its path. If the user does not find the document, the national level cache notifies the server directly.

Hierarchical architectures are more bandwidth efficient than other techniques, especially when some cache servers do not have high-speed connectivity.[7] But there are several problems associated with hierarchical caching.[8,9] For example, to set up a hierarchy, it may be necessary to place cache servers at key access points in the network, which requires significant coordination among participating cache servers.

In distributed caching, caches are only available at the bottom level, and the only intermediate cache levels are the institutional caches, which serve each other. All institutional caches keep data information about other institutional caches to use in deciding where to retrieve a document. Distributed caching directs traffic flows through less congested networks and facilitates load sharing and fault tolerance. Nevertheless, a large-scale distributed-caching deployment is likely to encounter several problems, such as high connection times, higher bandwidth usage, and administrative issues.

Hybrid caching combines hierarchical and distributed caching techniques. The hybrid scheme allows caches to use distributed caching to cooperate with other caches at the same level or at a high level. There has been some discussion about limiting the cooperation between neighbor caches to avoid obtaining documents from distant or slower locations.[10]

## Prefetching

Although caching improves performance, the benefits users can derive from caching will inevitably be limited. Previous research has shown that the maximum cache hit rate that any caching algorithm can achieve is usually less than 40 to 50 percent.[7,11]

Prefetching, on the other hand, increases the cache hit rate by anticipating the documents the user might want to retrieve. While the user views one document, a preemptive system downloads documents into the local cache for future viewing. A good prefetching scheme depends on a good prediction algorithm.

As with caching, prefetching also has disadvantages, which include increasing network resources and traffic use because of the additional (and sometimes overzealous) requests the system makes on the user's behalf. Prefetching only works well if the system can anticipate user requests without downloading everything on the Internet.

Early studies focused on prefetching schemes between browser clients and Web servers.[7] Other research used Web server traces and trace-driven simulation to study prefetching latency reduction.[12] This research showed that data prefetching from Web severs to individual clients can reduce client latency by 45 percent, but it does so at the expense of doubling the network traffic. However, these early studies did not address caching proxies so they don't completely answer the question about prefetching performance.

Security developers first used proxies to facilitate Web access and enhance security, but research has shown that using this approach for prefetching can reduce client latency. A proxy server typically processes requests from within a firewall by forwarding those requests to the remote servers, intercepting the responses, and returning the replies to the clients. One study evaluating the effectiveness of using proxies for prefetching techniques showed that combining perfect caching and perfect prefetching can reduce latency by 60 percent for high-bandwidth clients.[13]

Prefetching can also be used between browser clients and proxies. One approach is for the proxy to predict which documents a client might reference next and take advantage of the idle time between user requests to either push or pull the document to the user. A simulation study showed that prefetching combined with large browser caching and data compression can reduce client latency up to 23.4 percent.[14]

## Load-balancing techniques

Load balancing distributes the workload between servers and increases the efficiency of using specific features of certain servers. Round-robin DNS is a load-balancing technique that associates a domain with several IP addresses, each of which represents a different Web server. For a DNS lookup, the server returns the domain-IP mapping in a round-robin fashion, rotating each request to the next server in line. The round-robin technique is relatively easy to implement, and it does not require extra devices because most servers can be configured to use this DNS scheme. On the other hand, client-side caching causes some uneven traffic, and node failure is not uncommon in these environments. Finally, the very short time-to-live of the domain-IP mappings introduces extra DNS lookup traffic.

**Prefetching increases the cache hit rate by anticipating the documents the user might want to retrieve.**

In the L4 (or TCP router) load-balancing method, the L4 switch sits between Web servers and the Internet. Its IP address represents the address of the entire server farm. The switch forwards the traffic based on the TCP flows (IP source and destination addresses, TCP port number, and a certain bit in the TCP header). Packets in one TCP flow must be forwarded to the same server. The forwarding algorithm can use the round-robin technique or it can base its load balancing on contextual information, such as current load of servers or specific client IP addresses. Earlier research discusses various L4 switch architectures and several switching algorithms.[15] Compared to round-robin DNS, L4 switching is more flexible in distributing Web workloads and can more easily accommodate server failure.

An L7 switch also sits in front of the Web server. While an L4 switch is blind to HTTP requests, an L7 switch is aware of them. An L7 switch has even more flexibility than an L4 switch in forwarding Web traffic. The forwarding algorithm can use the session information or requested URLs other than TCP/IP-level information. Session-aware capabilities help send requests of one session to the same server to take advantage of previously negotiated information. L7 switches also can distribute requests on one TCP connection to different servers. However, this flexibility is gained at the price of the overhead associated with L7 switches, which are expensive and difficult to maintain.

## Content distribution network

A content distribution network is an architecture of Web-based elements arranged for efficient content delivery. CDNs typically distribute graphic files. They help push Web content as close as possible to users, take advantage of special servers to provide unique content, and balance loads between servers. CDNs use

**Table 2. Page sizes on the top-10 rated Web sites in the US before and after removing white space.**

| Web site | Size in Kbytes before removing white space | Size in Kbytes after removing white space | Percent change |
|---|---|---|---|
| AOL Time Warner network | 39.5 | 28.0 | 29 |
| Microsoft sites | 35.9 | 27.3 | 24 |
| Yahoo | 30.4 | 22.6 | 26 |
| Lycos | 24.8 | 19.0 | 23 |
| Excite network | 54.3 | 37.1 | 32 |
| About—The Human Internet | 42.9 | 29.1 | 32 |
| Walt Disney Internet Group | 10.1 | 8.1 | 20 |
| Infospace | 58.6 | 45.5 | 22 |
| CNET networks | 55.4 | 43.9 | 21 |
| Amazon | 62.6 | 45.8 | 27 |

manual hyperlink selection, HTTP redirection, DNS redirection, URL forwarding, and L4/L7 switching.

CDNs are an excellent way to speed information transfer. However, this relatively new technology is sometimes overlooked. The technology speeds Web page download times while reducing the need for costly new servers or additional bandwidth.

Cache servers within CDNs are placed all around the network and are arranged for efficient delivery of critical Web components. Acceleration of Web clients versus Web servers depends on the arrangement of cache servers within a network. CDNs are mostly used to cache image files and multimedia that does not change often over time. Introducing delta encoding will have a positive impact on CDN performance.

## Transmission schemes

Clients and servers have used HTTP 1.0 for a long time, but an increasing number have been moving to HTTP 1.1. The de facto use of HTTP 1.0 has required clients to open multiple parallel connections to the same server to improve performance over serialized connections. HTTP 1.1 supports persistent connections, which can be used either for serialized request-response pairs or for pipelining multiple requests and receiving multiple responses.

In HTTP 1.0, the server had to wait for a reply after sending every IP packet before sending another packet. In HTTP 1.1, the server can send an entire train of packets without waiting for a TCP acknowledgment. This pipelining technique reduces the total elapsed time between the initial request and the final reply without affecting the serial nature of the requests. By generating fewer but larger IP packets, HTTP 1.1 reduces network traffic and perceived download times.

## Efficient content

A page's size and complexity, including the number of items on it, are the most significant contributors to download time. Pages with a few simple items load the fastest and yield the most satisfied customers. Larger items always take longer to load but do not necessarily deliver more information or better function. The complexity of a page affects how quickly a server can present it. Factors that contribute to page complexity include large tables, dynamically generated table cells, JavaScript applications, or some other form of applet. The delays vary from browser to browser and from level to level within browsers.

You should only use multimedia effects when they truly add to the user's understanding. You can easily deliver rich layout using other techniques. Modularity allows you to apply the same style sheet to many documents, which can help reduce the need to send redundant presentation information.

Page designers often use white space to help them visualize their page presentations. White space in pages requiring encryption might actually, as a collective whole, add up to a lot of wasted bandwidth. While extra white space in clear text can be compressed well, encrypted white space does not compress well. Thus, to a certain extent, pages that use a lot of white space contribute to unnecessary network congestion.

We conducted an experiment using HTML Compress, a free tool from FreeSoft (http://www.freesoft.fsnet.co.uk/index.htm) that provides options for compressing or removing unwanted HTML code. However, we chose to only remove white space to observe its impact on file size. As Table 2 shows, this experiment demonstrated that removing white space can reduce the file size by 20 to 30 percent. Such a decrease in the size of HTML files has a noticeable impact on response time.

## Image optimization

Download speeds are an important consideration when deciding to add images to your Web content. When using a 56.6-Kbps modem, it would take about 20 seconds to download a 100-Kbyte image. This is far greater than the maximum 10 seconds permitted to maintain a user's interest. By choosing a more suitable file format, the same image can download in five seconds or less.

Graphics quality is based on the number of colors in an image. The more colors an image displays, the larger the file required to store it. Choosing 16.7 million colors instead of simple 256 colors increases the size of the graphic file by a factor of three. A 256-color image would require only one byte (8 bits) to represent the colors whereas 16.7 million colors require three bytes (24 bits).

You should avoid BMP file formats because they are uncompressed and thus have a relatively larger file size. Image files in GIF format use a nonlossy compression format and are more suitable for Web publishing, although they require higher rendering time. Another popular option is the JPEG format, which uses lossy compression and allows the user to balance quality with file size.

## Compression schemes

Using compression schemes can help reduce Web content, which decreases transmission time through the network. Compressed content also saves stored space in the cache. But the drawback to using compression schemes is the extra processing required on both the server and client sides. In the Delta content-coding scheme, clients only need to request that the server send the differences between two versions of a requested document, rather than the entire document.[12] This scheme helps save transmission time by reducing the size of the response message.

The future of Web performance improvements lies in developing additional techniques to help implement efficient, scalable, and stable improvements that enhance the end user experience. As Web services become increasingly popular, users continue to suffer from poor response times. Unfortunately the delays are cumulative, and it is not possible to decrease response time by improving a single slow link in the chain. Although several efforts have been made to improve Web performance, at the present time, the most viable solution is to favor simplicity in Web design by using a minimum amount of graphics and multimedia effects. ✸


### References

1. Zona Research, http://www.zonaresearch.com/infopress/99-jun30.htm.
2. J. Almeida, V. Almeida, and D. Yates, "Measuring the Behavior of a World Wide Web Server," *Proc. 7th IFIP Conf. High Performance Networking* (IFIP), Kluwer Academic Publishers, Norwell, Mass., 1997, pp. 57-72.
3. J. Banga and J. Mogul, "Scalable Kernel Performance for Internet Servers under Realistic Loads," *Proc. Usenix 1998 Technical Conf.*, Usenix, Berkeley, Calif., pp. 1-12.
4. J. Fielding et al., "Hypertext Transfer Protocol-HTTP/1.1 IETF RFC 2068," Jan 1997.
5. E. Cohen and H. Kaplan, "Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency," *Proc. IEEE Infocom 2000*, IEEE Press, Piscataway, N.J., 2000, Vol. 2, pp. 854-863..
6. C. Marsan, "VeriSign Unveils Managed DNS Service," *Network World*, Mar. 2001, http://www.nwfusion.com/news/2001/0313verisign.html.
7. J. Wang, "A Survey of Web Caching Schemes for the Internet," *Comm. ACM*, Oct. 1999, pp. 36-46.
8. P. Rodriguez, C. Spanner, and E.W. Biersack, "Analysis of Web Caching Architecture: Hierarchical and Distributed Caching," *IEEE/ACM Trans. Networking*, Aug. 2001, pp. 404-418.
9. R. Tewari et al., *Beyond Hierarchies: Design Consideration for Distributed Caching on the Inter*net, tech. report TR98-04, Dept. Computer Science, Univ. of Texas at Austin, 1998.
10. M. Rabinovich, J. Chase, and S. Gradde, "Not All Hits Are Created Equal: Cooperative Proxy Caching over a Wide-Area Network," *Computer Networks and ISDN System*, Nov. 1998, pp. 2253-2259.
11. M. Abrams et al., "Caching Proxies: Limitations and Potentials," *Proc. 4th Int'l World Wide Web Conf.*, W3C, Cambridge, Mass., 1995, pp. 119-133.
12. J. Mogul, "Delta Encoding in HTTP," Oct. 2000, http://search.ietf.org/internet-drafts/draft-mogul-http-delta-07.txt.
13. T. Kroeger, D. Long, J. Mogul, "Exploring the Bounds of Web Latency Reduction from Caching and Prefetching," *Proc. 1997 Usenix Symp. Internet Technologies and Systems*, Usenix, Berkeley, Calif., pp. 13-22.
14. L. Fan et al., "Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance," *Proc. ACM Conf. Measurement and Modeling of Computer Systems*, ACM, New York, 1999, pp. 178-187.
15. M. Colajanni, V. Cardellini, P.S. Yu, "Dynamic Load Balancing on Web-Server Systems," *IEEE Internet Computing*, May/June 1999, pp. 28-39.



***Mazen Zari*** *is completing an MS in electrical engineering and computer science at the University of Kansas. His research interests include networking and software engineering. He is a member of the IEEE Computer Society. Contact him at zarim@eecs.ku.edu.*

***Hossein Saiedian*** *is a professor and associate chair in the Department of Electrical Engineering and Computer Science at the University of Kansas. His research interests include software process, software architecture, object technology, and formal methods. He received a PhD in computing and information sciences from Kansas State University. He is a senior member of the IEEE and a member of the IEEE Computer Society, the ACM, and Sigma Xi. Contact him at saiedian@eecs.ku.edu.*

***Muhammad Naeem*** *is a software engineer at Sprint PCS in Overland Park, Kansas. His research interests include networking system performance and object-oriented programming. He received an MS in electrical engineering and computer science from the University of Kansas. Contact him at naeem01@sprintspectrum.com.*