World Wide Web Network Traffic Patterns

Jeff Sedayao

Intel Corporation, 2250 Mission College Blvd

Santa Clara, CA 95052

Abstract

The World Wide Web (WWW) generates a significant and growing portion of traffic on the Internet. With the click of a mouse button, a person browsing on the WWW can generate megabytes of multimedia network traffic. WWW's growth and possible network impact merit a study of its traffic patterns, problems, and possible changes. This paper attempts to characterize World Wide Web traffic patterns. First, the Web's HyperText Transfer Protocol (HTTP) is reviewed, with particular attention to latency factors. User access patterns and file size distribution are then described. Next, the HTTP design issues are discussed, followed by a section on proposed revisions. Benefits and drawbacks to each of the proposals are covered. The paper ends with pointers toward more information on this area.

1.0 Introduction

The World Wide Web [1] has been called the "killer app" of the Internet. Whole new businesses are being created to make advertising and information available on the Web. World Wide Web traffic is growing at the rate of over 20% PER MONTH [2]. With such an incredible growth rate, it is becomes critical for network engineers and technology managers to understand the impact of World Wide Web traffic on networks. In fact, some network administrators fear the World Wide Web because they fear its traffic implications.

This paper attempts to characterize the network impact of the Web. The first section reviews the HyperText Transfer Protocol (HTTP) [3], the protocol used by the World Wide Web. User access patterns are also covered. The next section examines the issues caused by HTTP and the use of the Web. The last major section of the paper describes current attempts to deal with those Web traffic issues.

2.0 World Wide Web Traffic Characteristics

To understand the World Wide Web's traffic characteristics, a brief understanding of the HyperText Transfer Protocol (HTTP), the protocol used by WWW, is necessary. HTTP is designed to be a simple request/response protocol. A client opens up a connection to the server, sends a request, gets a response, and then closes down the connection. Two key elements are the Uniform Resource Locator (URL) [4] and the method. The URL is a construct that provides information on the network location of some document on the Internet, including what server it lives on and how to access it. The method describes a specific action for a server to take.

The most common HTTP request (and the one we will focus on) uses the "GET" method. A client first sets up a connection (usually through TCP [5]) to the target Web server. Next, the client sends GET (the method) followed by a series of definitions of what data formats it will accept. The server processes the request and sends a "meta-description" of the document to the client, followed by the document itself. The document (also known as a page) can be a variety of things. It can be a form, an image, text, or a video or audio clip. After receiving the information, the connection is closed.

What happens on the packet level with a Get Request? Pamanabhan and Mogul [6] describe this in their study of HTTP. The client opens the TCP connection, resulting in an exchange of packets between client and server. That is one round trip. The HTTP request is then sent and documents received. This is the second round trip. For each URL retrieved, there are two round trips between the client and server.

WWW documents may contain “inline images”. These are images within a document. Web browsers process these images in the following way. First, they send a GET request for the document. Then they must send another HTTP GET request for each unique image. Thus a document (also known as a page) will see 2n + 2 packet round trips, where n is the number of unique images in that document.

This is only HTTP generated traffic. Another study of Web traffic [7] notes that Domain Name Service [8] (DNS) queries are also made on each HTTP request. A client needs to do a DNS name to address lookup on each URL. It does this to obtain the network address of the server it must access. The server, in turn, may look up the name associated with the client’s network address and then look up the name to verify that the name is associated with that network address. All three DNS requests can result in packet round trips between the client network and the server network. The following is a summary of the steps in a single HTTP request:

1. DNS name to address lookup.
2. Connection set up.
3. DNS address to name lookup.
4. DNS name to address lookup.
5. Send HTTP request and received page.

Each of these can result in packet round trips. Some, like steps 1, 4, and 5, can result in more than one packet round trips.

What are typical WWW access patterns? We looked at records of WWW activity at Intel over 32 weeks. The most common type of document or object is the Graphic Interchange Format [9] (GIF). This makes sense because GIFs are used as inline images in Web documents. The mean size of a GIF file is about 17005 bytes, and with a median size of 4513 bytes. GIF files also generated most of the WWW traffic into Intel, followed by MPEG files (a video format). MPEG files averaged 609146 bytes in size. This indicates a traffic distribution that is greatly skewed between many small objects (GIFs) and a number of very large objects (MPEG videos).

3.0 Traffic Issues

To discuss traffic issues, we need to define two terms - bandwidth and latency. Bandwidth is the amount of data that can be moved through a network link during a given time. It is usually described in units of bits/second. Latency is the time it takes data to travel from a given point in a network to another given point in the network. It is usually described in units of seconds or milliseconds. The lower limit to any network latency is governed by the speed of light. The network latency between San Francisco and New York will never be less than 13.7 milliseconds, now matter how big the network bandwidth on a connection between the two cities.

One of the things that a first time Web user notices is that when he clicks on an anchor (indicating a hypertext link), he never knows how long it will take to load that page. The page may be a short page, or it may be a long page filled with in-lined images. This is because Web Page authors rarely indicate the size of the page pointed to by the anchor. Many WWW users find it disturbing when they have to wait unpredictable amounts of time for pages to load. From a network perspective, the result is also a bursty unpredictable traffic mix. As described above, traffic varies between small objects (mostly) and very large objects.

The HTTP protocol has a number of problems. It was designed to be as stateless as possible. GET requests are treated independently by an HTTP server. This makes HTTP servers easier to write. Unfortunately, this also brings in a number of inefficiences and leaves HTTP performance dependent on network latency. For each HTTP request, a minimum of two round trips is necessary. With worse case DNS lookups, up to five round trip times could be needed for the HTTP request. Spero’s analysis of HTTP [10] concludes that the two Round Trip Times (RTT) form the lower bound for the total transaction time of an HTTP request. No matter how big the network bandwidth is between client and server, the network latency between client and server determines minimum time to process a request.

Inline images make the problem even worse. For each page with inline images, a separate HTTP request must be made for the page and then for each unique image. Thus a single document could have from 1 to any number of HTTP requests associated with it. The minimum time to

load the document would be (2n + 2) * latency, where n is the number images in the page. This can also be expressed as (n + 1) * RTT.

A TCP feature called "slow-start" [11] causes inefficient use of available network bandwidth. Slow start is a feature of TCP that causes the amount of data sent to be small and then increase. A TCP connection will first send a small amount of data. As acknowledgements of packets come in, the window gets bigger and bigger. Unfortunately, most HTTP connections are very short lived. The most common object is a GIF file (median size of 4513 bytes). Many TCP connections used in HTTP will not be at full throttle before they are terminated.

While GIFs generate the most traffic, MPEG video clips generate the next most traffic. This skewing of traffic between lots of small images and fewer but very large video clips presents a difficult challenge for network administrators. HTTP wants both low latency yet high bandwidth networks. Network bandwidth can be increased (often at very high cost), but there are absolute lower bounds on latency.

A final inefficiency involves TCP protocol states on the server. TCP specifications require that a system that has closed a connection maintain connection information for four minutes [5]. The large number of connections could cause a server to have its connections table filled with "TIME-WAIT" state connections.

4.0 Proposed Solutions

The problems described above have been widely discussed. A number of solutions have been proposed. This section discusses the pro's and con's of a number of solutions.

4.1 Be Careful and Smart with Cache

This solution [7] proposes that Web page authors, Network administrators, and WWW browser authors do a number of things to minimize the number of TCP connections necessary and thus minimize transaction time. Web authors can keep in-lined images to a minimum and make pages that are cachable. Network administrators can set up Proxy Caching Servers [12]. Proxy Caching Servers get Web information on behalf of clients. They cache pages, so that if a page or URL has been requested before, a client can immediately get the cached page rather than waiting to get it from the Internet. It is also a good idea to deploy caching DNS servers on or near the caching servers. This will reduce the latencies involved with DNS lookups. Web Browsers can cache pages and images. Many browsers do just that, reducing the amount of network traffic and time needed to load a page.

These ideas have a number of benefits. They can be done immediately and effectively with existing software and protocols. There are a few drawbacks to consider. Caching doesn't work for dynamically changing data like stock quotes (stock quotes are a very popular use of WWW) Also, these strategies don't address the fundamental problems with HTTP. Items that are not cached will still be affected by network latencies and will experience extra delay caused by looking through a cache and going through the proxy server.

4.2 Parallel GETs

Another performance-enhancing tactic is for browsers to send out GETs for inline images without waiting for the initial GET to complete. The GETs are basically processed in parallel as data comes in. This is a big advantage over browsers that wait for each in-lined image to complete before getting another one. But like the previous idea, it doesn't deal with the fundamental problems of HTTP. Total transaction time will still have a lower bound of (n + 1)*RTT , where n is the number of images in a document. In fact, it can make matters worse for Web servers and proxy caching servers. Many server implementations spawn a process for each URL. Instead of processes being smoothly created one after another, large batches of processes are created almost simultaneously. This process burstiness can really impact a server.

4.3 URNs

Uniform Resource Names (URNs) [13] are constructs under development by the Internet Engineering Task Force (IETF). They are unique and permanently assigned names for resources. URNs map to a number of URLs. A browser or proxy server could permanently cache a number of URNs. It potentially could look for and access the URL that had the smallest network latency.

URNs are coming, and they will offer better caching performance. Still, for uncached URNs, the HTTP protocol problems have not been solved. Doing the URN to URL mappings will add latency to HTTP requests. Also, while URNs are coming, the infrastructure for resolving URNs is not yet available.

4.4 GETLIST and GETALL

Pamanabhan and Mogul [6] propose two new methods. GETLIST would get a list of URLs. GETALL would get all the URLs (images) in-lined in a page. These two methods solve the connection problem by transferring all the needed URLs in one connection. If images were already cached, the GETLIST method would be used to get only the images needed. These two methods would create longer lived connections and reduce both the number and the relative cost of setting up a connection. Some drawbacks to this approach are that changes in existing WWW servers and clients would have to be made. Also, adding these commands would make clients and servers more complex, as much more state would have to be managed.

4.5 HTTP Session Layer

Spero [14] proposes adding a session layer to HTTP in a new protocol called HTTP-NG (HTTP Next Generation). This session layer would divide a connection into different channels. Control messages (HTTP requests and meta-information) would flow over a control channel. As in the previous suggestion, this proposal solves connection problems by adding complexity and state to clients and servers. There would also be issues with transition, although Spero proposes a solution using intermediary proxy servers. Other people are working on session layer based solutions [15].

4.6 MIME Multipart Documents

HTTP wraps data objects in the MIME [16] multimedia mail format. One suggestion for eliminating the multiple connection problem is to transfer a document and all its images in a multipart MIME type. This could be done in one connection. This is an elegant solution, but like all the solutions, it has a few drawbacks. While WWW servers and browsers are already supposed to be capable of doing this, few of the popular browsers and servers are. Two other problems with this scheme need to be considered [17]: (1) MIME encoding will roughly double the number of bytes used sent and (2) by loading in a complete document, there is no way to stop already cached images or documents from being reloaded (unlike the GETLIST/ GETALL proposal).

5.0 Conclusion

HTTP and patterns of World Wide Web use create a number of challenges to network infrastructure. This paper has covered WWW traffic patterns, the issues raised, and possible solutions to those problems. Work on this issue is ongoing. One of the best places to monitor WWW traffic developments is on the Web itself. In particular, discussions on solving HTTP problems can be examined at the HTTP-WG mail archives (URL http://www.ics.uic.edu/ pub/ietf/http/hypermail).

6.0 References

[1] Tim Berners-Lee, R. Cailiau, A. Luotonen, H. Nielsen, and A. Secret. The World Wide Web. *Communications of the ACM*. 37(8):76-82, August 1994.

[2] Tony Rutkowski. *Internet Traffic*. URL ftp:// ftp.isoc.gov/isoc/charts/traffic4.ppt, December 10, 1994.

[3] Tim Berners-Lee. *Hypertext Transfer Protocol (HTTP)*. Internet Draft draft-ietf-iir-http-00.txt, IETF. Novermber, 1993. This is working draft.

[4] T. Berners-Lee, L. Masinter & M. McCahill. *Uniform Resource Locators (URL)*. RFC 1738. December, 1994.

[5] Jon. B. Postel. *Transmission Control Protocol*. RFC 793. September, 1981.

[6] Venkata N. Padmanabhhan and Jeffrey C. Mogul. Improving HTTP Latency. *Proceedings of the Second International World-Wide Web Conference*, pages 995-1005, Chicago, October 1994.

[7] Jeff Sedayao. Mosaic will kill my Network! *Proceedings of the Second International World-Wide Web Conference*, pages 1029-1038. Chicago, October 1994.

[8] P. Mockapetris. *Domain names - concepts and facilities*. RFC 1034. November 1987.

[9] CompuServe, Incorporated. *Graphic Interchange Format Standard*. 1987.

[10] Simon E. Spero. *Analysis of HTTP Performance Problems*. URL http://elanor.oit.unc.edu/http-prob.html, July 1994.

[11] Van Jacobsen. Congestion Avoidance and control. *Proceedings of SIGCOMM '88 Symposium on Communications Architectures and Protocols*. pages 314-329. Stanford, CA, August 1988.

[12] Kevin Altis and Ari Luotonen. World Wide Web Proxies. *Proceedings of the First International World-Wide Web Conference*. Geneva, April 1994.

[13] K. Sollins, L. Masinter. Functional Requirements for *Uniform Resource Names*. RFC 1737. December 1994.

[14] Simon E. Spero. *Progress on HTTP-NG*. URL http://www11.w3.org/hypertext/WWW/Protocols/HTTP-NG/http-ng-status.html.

[15] Dave Raggett. *Minutes from the December 1994 San Jose IETF (HTTP BOF)*. URL http://www.ics.uci.edu/pub/ietf/http/minutes-SJ.txt

[16] N. Borenstein and N. Feed. *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*. RFC 1521. September 1993.

[17] Mitra. "Re: HTTP: T-T-T-Talking about MIME Generation". URL http://www.ics.uci.edu/pub/ietf/http/hypermail/