Analysis of HTTP/1.1 Performance on a Wireless Network

Stephen Cheng Kevin Lai Mary Baker
{stephenc, laik, mgbaker}@cs.stanford.edu http://mosquitonet.stanford.edu

Techical Report: CSL-TR-99-778
February 1999

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305-9040

This research is supported by a gift from NTT Mobile Communications Network, Inc. (NTT DoCoMo), a graduate fellowship from the USENIX Association, and a Sloan Foundation Faculty Fellowship.

Abstract

We compare the performance of HTTP/1.0 and 1.1 on a high latency, low bandwidth wireless network. HTTP/1.0 is known to have low throughput and consume excessive network and server resources on today's graphics-intensive web pages. A high latency, low bandwidth network only magnifies these problems. HTTP/1.1 was developed to remedy these problems. We show that on a Ricochet wireless network, HTTP/1.1 doubles throughput over HTTP/1.0 and decreases the number of packets sent by 60%.

I. Introduction

In the past few years, there has been an explosion in the popularity of the World Wide Web. The convenience of having information immediately available is indispensable. Users are able to connect to computers in all parts of the globe from their home or office. However, as the number of Web users has increased, so have the number of problems. One problem is the performance and impact on the network of HTTP/1.0, the underlying protocol for transferring Web pages. HTTP/1.1 was developed to address this problem. It aims to 1) reduce traffic on the network by only opening one persistent network connection instead of many short-lived ones, 2) improve throughput by pipelining requests, and 3) improve caching. HTTP/1.1 is backward compatible with HTTP/1.0 and is gaining wider acceptance as Netscape and Microsoft build support for it into their browsers.

The interest in the World Wide Web is only a small part of the overall trend towards greater connectivity that has permeated society. At the same time as the Web has allowed people to contact data all over the world, metropolitan area wireless and cellular networks have allowed people to remain in contact wherever they go. However, the high latency and low bandwidth of these networks have limited the freedom of users in accessing data on the Web. These networks often have high latency because they use latency-increasing link level error recovery techniques and packets often have to traverse multiple wireless hops before entering the wired part of the Internet.

Many of the networking problems which are addressed by HTTP/1.1 are magnified in a high latency network such as a wireless link. Reducing the number of packets sent and mitigating the effect of latency through pipelining would benefit a wireless network even more than its wired counterpart. We investigate the effects of HTTP/1.1 on a Metricom Ricochet wireless network and compare it with the effects on an Ethernet network.

II. HTTP

In this section we discuss the major problems of HTTP/1.0 and how HTTP/1.1 solves those problems.

A. Problems with HTTP/1.0

The primary problem with HTTP/1.0 is that it opens a separate connection for each object on a page [4]. Each image on a page is usually a different object and therefore HTTP/1.0 usually has to open many connections for each page.

Opening many connections for a page has a number of detrimental effects on throughput and the network load. First, each connection requires a certain amount of time for setup and shutdown. This decreases the throughput experienced by the user. In addition, the packets used for setup and shutdown don't contain application data, thereby unnecessarily increasing network load.

Second, each connection consumes state in the HTTP server. HTTP uses TCP for its transport protocol. TCP specifies that a connection must remain in a waiting state for four minutes after it is closed. Thus, a server may be littered with many ports in this state. In many TCP implementations, the more ports a server has open, the longer it takes the server to determine which application a packet is destined for. We didn't measure the effect of excessive state on the HTTP server because it would be the same regardless of whether we were using a wired or wireless network.

Another detrimental effect is that TCP cannot utilize the full bandwidth of the network. Most HTTP/1.0 connections are very short, since most HTTP objects are small. TCP has a slow start mechanism which gradually increases the sending rate in order to probe the network for the best throughput rate without causing congestion. However, a short-lived connection may never reach its full throughput potential. This causes the throughput experienced by the user to be unnecessarily low.

Another problem is that HTTP/1.0 has poor throughput on high latency connections because it is a strict request/response protocol. This means it requests an object and then waits until it has received the object before requesting the next object. As a consequence, HTTP/1.0 requires at least two network round trips for each retrieved object, which greatly decreases throughput on high latency networks such as the Metricom Ricochet.

The final problem is that HTTP/1.0 usually causes high internal fragmentation (not the same as IP fragmentation). This is because most HTTP objects are small and HTTP/1.0 uses a separate connection for each object. For example, an object slightly larger than a packet will cause one full-size packet and one small packet to be sent because we can't pack data from another object in with the last bit of data from the first object.

High internal fragmentation causes the many small packets to be sent. Small packets are wasteful of network resources because packets have a fixed cost of certain network resources regardless of the size of the packet. For example, routers have to spend some amount of CPU time to process a packet regardless of size. For a small packet, this CPU time cannot be amortized over many bytes, and therefore the router isn't working at optimal efficiency. Another example is the bytes used by packet headers. HTTP packets are also TCP packets and therefore have a header of 40 bytes per packet. The smaller the packet size, the greater the relative cost of the header in bandwidth.

B. HTTP/1.1

HTTP/1.1 was developed to address these problems. It is backward-compatible with HTTP/1.0 but manages connections differently. Instead of making a separate connection for each Web object, an HTTP/1.1 client makes a single *persistent* connection with the server, which it leaves open for subsequent retrievals, thus solving the above problems:

1. The client doesn't need to waste time or packets setting up multiple connections.
2. The server doesn't need to maintain state for those extra connections.
3. The single HTTP/1.1 connection transfers more data than each of the HTTP/1.0 connection and therefore can more effectively use slow start to fully utilize the network's bandwidth.
4. HTTP/1.1 *pipelines* its requests, enabling the client to issue multiple requests before receiving any responses from the server.
5. Since the client uses pipelined requests over one connection, different objects sent from the server can be packed into a fewer number of packets, resulting in lower internal fragmentation.

Having a single *persistent*, *pipelined* connection allows the HTTP/1.1 to reach a throughput approaching the bandwidth of the network while only sending the minimal number of packets.

III. Experiment Setup

In this section, we discuss the hardware and software we used to run the experiments.

We used a relatively simple networking setup. We had two Pentium computers, `tnt.stanford.edu` and `detcord.stanford.edu`, running RedHat Linux 5.0 (kernel version 2.0.33). Each machine had a wired and wireless connections and two IP addresses. The wireless connection employed Metricom Ricochet SE radios (Firmware version 106503-210P), and the wired connection was a 10-base-T Ethernet network.

The Ethernet network has an Maximum Transmission Unit (MTU) of 1500 bytes, a bandwidth of 10Mb/s, and an average round trip time (RTT) of 4 ms. The Ricochet network has an MTU of 1152 bytes. We used the Ricochet network so that the radios talk peer to peer. In this configuration, the Ricochet network has a bandwidth of 60Kb/s and an average RTT of 170ms. This is not the configuration seen by most Ricochet users (see section VI), but it provides the maximum possible performance for the Ricochet network and avoids the error introduced by sharing bandwidth with other users during the tests.

We spent most of our time configuring our software. We used Apache 1.2.6 [1] as our web server and the libwww robot (release 5.1l) from the W3 Consortium [2] as our client.

The Apache server was not compiled with any special options, other than those required to get it running on RedHat 5.0. However, in the runtime configuration files, we placed in an option to make it emulate an HTTP/1.0 only server. We set the special environment variable "downgrade-1.0" to true regardless of the browser. Thus, we had to start up the server twice: once to gather HTTP/1.0 data and once to gather HTTP/1.1 data. We decided to force the server into HTTP/1.0 mode rather than the robot because we were unable to get the robot to perform correctly in the mandatory HTTP/1.0 mode. We set up a sample web page which was basically a copy of a Microsoft web page. It was meant to simulate current web pages which are graphically intense. The HTML file itself was approximately 25KB and it contained 30 in-line images varying from under 100 bytes to about 10KB. The total amount of data (text and images) was about 69KB.

The robot was run in non-interactive mode, saving the images in order to simulate a GUI browser. In addition, the quiet mode was turned on so that the output would not slow down the time measurements.

	HTTP/1.0 Wired	HTTP/1.1 Wired	HTTP/1.0 Wireless	HTTP/1.1 Wireless
Throughput (Kb/s)	247.6(4.7)	462.4(18.9)	17.2(15.5)	36.9(12.4)
Total Packets	363.6 (0.4)	95.9 (4.2)	416.7 (2.5)	170.2 (5.5)
Packet Size (bytes)	304.2 (0.6)	947.6 (1.0)	276.3 (3.2)	578.48 (5.5)

TABLE I

Performance of HTTP/1.0 and 1.1 on Ethernet and Ricochet. This table lists the mean and percent standard deviation for ten runs.

To time the duration of the connections, we simply took the time elapsed value printed out by the robot. Because the time given by the robot includes the time taken to start up the robot and exit the program, these times are not the exact times between when the first SYN request was sent and the last ACK received. However, the overhead of the robot is small compared to the network events were measuring, so we ignored this source of error.

A more serious source of error is the fact that the robot makes one TCP connection per request and one request at a time in HTTP/1.0 mode. We were unable to do parallel connections to mimic browsers like Netscape (which typically uses four parallel connections). In HTTP/1.1 mode, the robot makes one TCP connection for all requests using the persistent connection protocol and pipelining. We did not use the persistent cache because we are simply measuring the effects on the transfer of TCP packets, not necessarily overall web performance.

We gathered the data using the robot's quiet mode output and using `tcpdump` on the server side. We started the Apache server (in HTTP/1.0 mode) and then `tcpdump` on `tnt`. On `detcord`, we started the robot with the IP address of the server's wired interface. We then ran the again, this time with the IP address of the server's wireless interface. Then we killed the the server and restarted it with the edited configuration file allowing normal HTTP/1.1 operation. The tests were repeated to get measurements for the HTTP/1.1 specification on both on the wired and wireless interfaces. We ran the test five times for each of the four different setups.

IV. Experimental Results

In this section we discuss the results of our experiments. We used `tcpdump` to dump the packet traces to a file and `tcptrace` [3] to analyze them.

A. Quantified Benefits of HTTP/1.1

Table I shows the mean and standard deviation over ten runs of various metrics which quantify the benefits of HTTP/1.1 over HTTP/1.0 over a wireless network. The standard deviation is given as a percentage of the mean. We also give the results over a wired network for comparison.

The first metric we are interested in is throughput. Throughput gives a first order approximation of the delay experienced by the user. HTTP/1.1 approximately doubles throughput over HTTP/1.0 on both the wired and wireless interfaces. HTTP/1.1 throughput is improved for many of the reasons mentioned in section II-B: no time is spent setting up extra connections, the longer connection time enables TCP to more fully utilize the network's bandwidth, pipelining of requests reduces the overall delay, and the lower internal fragmentation reduces bandwidth spent on headers.

The second metric is total packets sent into the network. We list this in the table as "Total Packets". This metric measures the network resources consumed by HTTP, thus indicating how well the network would scale to many users. We found that HTTP/1.1 decreased the number of packets sent by a factor of 3.8 on the wired network and 2.4 on the wireless network. This would represent a significant increase in the number of hosts which could use the network.

There are several reasons why HTTP/1.1 sends so many fewer packets than HTTP/1.0. As discussed in section II-B, HTTP/1.1 doesn't waste packets setting up connections because it only sets up one connection. Another reason is that HTTP/1.1 packets achieve a much lower internal fragmentation by sending fewer, but

	HTTP/1.0 Wired	HTTP/1.1 Wired	HTTP/1.0 Wireless	HTTP/1.1 Wireless
Number of runs	10	2	3	1
Total time	2.91 (4.7)	1.14 (0.6)	38.34 (7.9)	16.45 (0.0)
Packets (client to server)	168.5 (0.7)	25.5 (2.8)	193.33 (0.8)	62 (0.0)
Packets (server to client)	195.1 (0.3)	64 (0.0)	223.67 (0.3)	92 (0.0)
Total packets	363.6 (0.4)	89.5 (0.0)	417 (0.4)	154 (0.0)
Payload Bytes	90065.9 (0.7)	84432 (0.0)	91443.67 (0.4)	84711 (0.0)
Total Bytes	110620.7 (0.6)	89386 (0.0)	114201.67 (0.4)	93035 (0.0)

TABLE II

Performance of HTTP/1.0 and 1.1 on Ethernet and Ricochet, excluding runs where there were retransmissions. This table lists the mean and percent standard deviation for runs which didn't have retransmissions.

larger packets than HTTP/1.0. In table I we can see that the average packet size of HTTP/1.1 is almost three times larger for the wired network and twice as larger for the wireless.

B. Inconsistencies and Sources of Error

In this section, we discuss inconsistencies and sources of error in our data. The most significant source of error is that the throughput improvement described in section IV is greater than the actual improvement we would expect in browser performance. This is because (as noted before) we were unable to get our client to open more than one TCP connection simultaneously (as current browsers do).

We expect that this technique of opening multiple TCP connections simultaneously would give large performance improvements on the wired network. This is because short lived HTTP/1.0 TCP connections are never able to increase their windows to the point where they are able to fully utilize the 10Mb/s of bandwidth. This might even boost the throughput of HTTP/1.0 over that of HTTP/1.1. On the other hand, we expect that this technique would give little performance improvement on the wireless network because even short TCP connections are able to open their TCP windows to the point where they can use the 60 Kb/s maximum bandwidth of the Ricochet (see section III).

One inconsistency in data is the is the difference in total packets sent using HTTP/1.1 on the wired (mean of 95.9 packets) compared to the wireless networks (mean of 170.2 packets). At first glance, one would think that there should be no difference because the actual HTML and picture data sent in each case is the same.

There are several sources for this difference. One is retransmissions. If packets have to be retransmitted on the wireless network, this could account for the extra packets. In order to measure this effect, we recalculated our metrics using only those connections which experienced no retransmissions. The results are summarized in Table II. In this table we list the number of runs we could use to compute each column, since it might be different from the ten runs we used in Table I. We can see that when we factor our retransmissions, there is less difference (89.5 and 154) in the total packets sent into the network.

Another source for this difference is the different MTUs for the different networks (mentioned in section III). As mentioned in section II-B, HTTP/1.1 reduces internal fragmentation by filling up packets with more data. However, since the maximum packet size 1500 bytes for Ethernet is 1500 bytes and 1152 bytes for Ricochet, it needs to send more packets over Ricochet.

V. Related Work

Nielsen, et. al., [5] describe the performance improvements of HTTP/1.1 over HTTP/1.0 on a variety of network technologies, but none of them are wireless. They do measure the performance of HTTP/1.1 on a high latency, low bandwidth network (a modem), but do not measure the performance of HTTP/1.0 for comparison.

VI. Future Work

In this section, we discuss possible future experiments related to this research. One experiment would be measuring the same metrics, but using a client that supported multiple simultaneous open connections. This would accurately simulate the behavior of current browsers.

Another experiment would be to configure the wireless Ricochet network so that multiple wireless hops separate the client and the server. This is the configuration seen by most users of the Ricochet network.

In addition, the effect of congestion on HTTP/1.1 as not been thoroughly studied. We would like to study this as well as how these effects change on wireless networks. We would most likely have to use a network simulator to accurately measure these effects.

A network simulator would also allow us to measure the performance of HTTP/1.1 on wireless networks other than Ricochet. This would allow us to test hypotheses we might have about the performance of HTTP/1.1 on networks other than Ricochet. For example have the hypothesis that the throughput of HTTP/1.1 would scale with greater network bandwidth. The average throughput of HTTP/1.1 on Ricochet that we measured was 36.9Kb/s, which approaches the maximum bandwidth of 60Kb/s of the Ricochet network. This suggests that the throughput of HTTP/1.1 would scale with greater network bandwidth.

VII. Acknowledgements

We'd like to thank NTT DoCoMo for funding this research.

VIII. Conclusion

Our study has shown that on a wireless network, HTTP/1.1 doubles the throughput of HTTP/1.0, while reducing the number of packets sent by 60%. We believe this significantly improves the performance seen by the user and increases the scalability of the wireless network.

In the future, wide area wirless networks are likely to remain high latency because of the inherent interference wireless networks must overcome. In addition, for these wireless networks to remain inexpensive, there will have to be many wireless routers without a wired connection to the Internet, thus causing packets to traverse several high latency wireless hops. Similarly, satellite-based wireless networks are likely to have high latency. However, these wireless networks will have greatly increased bandwidth because of more efficient use of the spectrum and the allocation of more parts of the spectrum. In such an environment, we believe that HTTP/1.1 will have even greater benefits than we have measured.

References

[1] Apache. http://www.apache.org/, 1998.
[2] libwww robot. http://www.w3.org/, 1998.
[3] tcptrace. http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html, 1998.
[4] Jeffrey C. Mogul. The case for persistent-connection http. Technical Report Western Research Laboratory Research Report 95/4, Digital Equipment Corporation, 1995.
[5] Henrik Frystyk Nielsen, Henrik Frystyk, Jim Gettys, Anselm Baird-Smith, Eric Prudhommeaux, Hakon Wium Lie, and Chris Lilley. Network performance effects of http/1.1, css1, and png. In *Proceedings of SIGCOMM*, 1997. http://www.w3.org/Protocols/HTTP/Performance/Pipeline.