

Analysis of Sources of Latency in Downloading Web Pages

Md. Ahsan Habib (mhabib@vt.edu)

Marc Abrams (abrams@vt.edu)

Network Research Group (www.cs.vt.edu/~nrg)

Computer Science Department, Virginia Tech

Blacksburg, VA 24061-0106

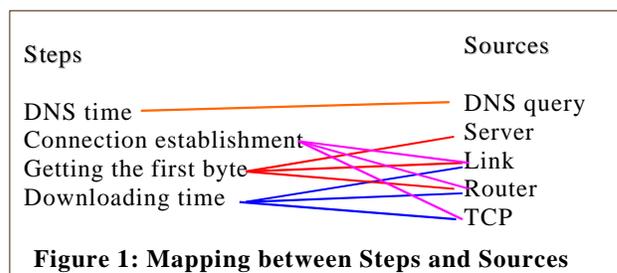
Abstract

Why does it take so long to download a Web page from a Web server? We analyze the download latency for pages for a variety of situations, in which Web browser and server are both within the same country as well as in different countries. Our study examines several sources of latency in accessing Web pages: DNS, TCP, the Web server itself, and the network links and routers. We divide the total download time into four parts: DNS query, connection setup time, time to get the first byte of a Web page, and downloading time. In most cases, roughly half of the time is spent from the moment the browser sends the acknowledgement completing the TCP connection establishment until the first packet containing page content arrives. The bulk of this time is the round trip delay, and only a tiny portion is delay at the server. This implies that the bottleneck in accessing pages over the Internet is due to the Internet itself, and not the server speed (as suggested by another study). The second bottleneck is the 3-way TCP connection establishment (consuming 1/5 to 1/4 of the delay). Conclusions are drawn on how to decrease latency.

Keywords: Bottleneck, Latency, DNS Query, Server Delay, HTTP, TCP

1 Introduction

Internet users who want to download a Web page from a remote machine (inside as well as outside the country) often experience poor performance. What causes poor performance? To investigate the causes, we have to examine the steps involved in downloading a Web page and the sources of bottleneck related to the steps. Steps involved in downloading a Web page are DNS query, connection establishment, waiting for the first byte, and downloading the page. The sources related with the steps can be potential candidates for bottlenecks, and those are DNS query, server, links, routers, and TCP. The mapping between the actions and sources are shown in Figure 1. We discuss each source briefly below



The first possible source of a bottleneck is a DNS query. When a page is requested in the Web, first the hostname is resolved to an IP address by DNS and then a connection is established from the client machine to the requested host. In presence of a proxy, the page may come from the proxy not from the host. We investigate the degree to which the DNS query time can be a bottleneck. What percentage of total access time is used for DNS query (in the cases of both a DNS cache hit and a miss)? How does DNS query time vary for domestic versus international sites? These are the basic questions addressed in this paper.

Second, the server can be a potential source of bottleneck and can cause unexpected delay to return a Web page. The server delay is mainly due to processing the client's request. So, the server delay depends on with several items, including server specification (CPU speed, memory), the server load, rate of requests arriving at the server. In our study, we vary the server load, rate of requests to measure whether or not the server delay is a significant part of the total downloading time.

After server, let's take a look on the path from client to server. This has consists of link and routers. So, the next item to be considered as a possible source of bottleneck is link. Bottlenecks due to links in an Internet route depend on the link properties. Obviously, the physical media (e.g., fiber, satellite) and protocol (e.g., FDDI, ATM) limit the RTT and maximum bandwidth of a link. The throughput of router also dominates the delay caused by the link.

The fourth possible candidate for bottleneck is a router. There are several reasons why a router may be a bottleneck. If a router has insufficient buffer space or processing speed, then it will discard packets. Packet loss will ultimately lead to a high RTT. The router throughput (e.g., number of packets per second that can be routed) may cause a bottleneck. Sometimes routing algorithms may be a headache, especially in international routes. An end to end routing behavior in the Internet is discussed in Paxson [12].

The fifth candidate for bottleneck is the transmission protocol used to download a Web page. TCP is used for HTTP and FTP in downloading Web pages. TCP has some properties that are not suitable for HTTP. TCP needs three-way handshaking for connection establishment and four-way handshaking for connection release. This handshaking causes great overhead in case of a small file. TCP has slow-start property that limits the number packet can be sent at the beginning of file transferring. To overcome all the above-described problems, we have to design a new transmission protocol for HTTP. In our study, we address another problem of TCP. As TCP is a reliable transmission protocol, it needs each packet to be acknowledged. TCP sets a timer after sending a packet. If the timer expires before getting the ack, TCP needs to retransmit the packet. Our concern is whether or not TCP incorrectly guesses the timeout value for a packet. If so, it will create extra latency due to unnecessary retransmission. Then overall downloading time will be high. In this paper we try to figure out how often TCP makes unnecessary retransmissions and whether or not we can save them.

In this paper, we go through all the above sources to identify bottlenecks and their causes.

2 Related Work

Our main question in this work is “why is the Web so slow?” This work is based on a number of previous works. We review here different works on the HTTP performance problem, the

problem of using TCP for accessing Web pages using HTTP, Web characterization, wide area network performance and active/passive network measurement.

Several authors have investigated the performance of HTTP. Spero et al [17] analyze the performance problem of HTTP. In Spero's [17] paper the problem with HTTP is explained by looking at the network traffic generated by a typical HTTP transaction. We also follow their steps of tracing HTTP traffic using *tcpdump* [6]. Spero et al [17] shows the performance problem of HTTP due to connection establishment, slow-start, open new connection per transaction and requesting single object per connection. It concludes, "HTTP/1.0 interacts badly with TCP. It incurs frequent round-trip delays due to connection establishment, performs slow start in both directions for short duration connections, and incurs heavy latency penalties due to the mismatch of the typical access profiles with the single request per transaction model."

Padmanabhan et al [15] examine improving HTTP latency. They modify HTTP/1.0 to avoid separate TCP connection for each file requested to access web page. They propose "GETALL", "GETLIST" methods to make pipeline requests which eventually reduce latency for a page that has multiple images.

Touch et al [16] also analyze the performance of HTTP. They discuss "the performance effects of using per-transaction TCP connection for HTTP access, and proposed optimizations of avoiding per-transaction re-connection and TCP slow-start restart overheads." Transaction TCP provides transaction-oriented service over TCP via extensions to the TCP protocol. HTTP performance can be improved by using persistent connections, in which a single connection can access multiple file [14]. Touch et al [16] suggest that "the persistent connection optimizations do not substantially affect Web access for the vast majority of users. Most users see end-to-end latencies of about 250 ms and use modem lines. Bandwidths over 200 Kbps are required to provide user-noticeable performance improvements." This conclusion is similar to the conclusion of our study. Heidemann [19] analyze the performance interaction between persistent HTTP (P-HTTP) and TCP implementations. Initial P-HTTP implementation has problems such as "TCP delayed-acknowledgements" and "multiple slow-starts per TCP connection". Solutions given by Heidemann [19] that makes P-HTTP performance better than standard HTTP. But according to

Bernardo [21], on average a user accesses just one URL at a site, which negates the benefit of using persistence.

Thompson et al [20] developed a monitoring system to capture and analyze Internet traffic on OC-3 trunks within internetMCI's backbone and also within vBNS. They reveal the characteristics of the traffic in terms of packet sizes, flow duration, volume and percentage composition by protocol application, etc. Thompson's [20] paper help us to understand domestic as well as international (US to UK and UK to US) traffic flow, their pattern and composition based on protocol application. According to Thompson [20], Web traffic dominates among applications making up 75 percent of overall bytes and up to 70 percent of overall packet flows during daytime. These numbers signify the importance of the problem of improving latency in downloading Web pages.

In order to assess the quality of Internet service, an ongoing Bellcore measurement site [1] has drawn at top 100 URLs, visited the reference to get the list, and measured the delay to wait for loading the page in four components (DNS delay, connection delay, server delay and transmission delay). They also seek to identify the Web bottlenecks. The first part of our study has some similarities with Bellcore [1] but our results differ with them. On Monday February 8 1999 at 15.33 test, the average server delay is 3.28 sec whereas the DNS delay is 1.84 sec and connection delay is 1.15 sec. They conclude that most of the delay in downloading Web pages is due to the server, which differs from our study. We explore all possible sources that can introduce latency in accessing the Web and causes of the latency.

Barford et al [2] measure “the Web performance in the wide area.” In their Wide Area Web Measurement (WAWM) project, they treat the server and client as an integrated system. They present performance measurements for file transfers over combinations of server load, network load, and file sizes. They measure latency and packet loss for different file sizes, under different server and network load (low/heavy). They have shown that “the main effect of server load on typical transfers is to delay the first data packet sent” and “servers under high load suffered significantly less packet loss than those under low load.” According to their paper, they want to use WAWM infrastructure to address the same above question "why is the Web so slow?" in future.

3 Bottleneck in DNS Query or Server

In this part of our study, we instrument the time taken of each step in accessing Web pages. The steps are DNS query, connection establishment, time to get the first byte, and the downloading time. The terms can be described briefly as follows

DNS Time: The DNS component measurement is the time spent resolving the DNS name to an IP address. Typically, this is measured from a DNS server located near the client that wants to communicate with a server.

Connection Setup Time: The connection setup measurement is the time required setting up a connection from a client to a Web server. This delay is the time needed to establish the TCP connection using 3-way protocol handshake.

Time to Get the First Byte: The Time to First Byte measurement is the amount of time from when the client sends the request (GET command) until it sees the first byte back from the server. This is a single round trip across the Internet.

Downloading Time: This measurement starts when the first byte arrives and ends when the last byte of the file arrives to the client. We normalize the downloading time to the time needed to download 1KB file by dividing download time by file size in multiples of 1024. HTTP header bytes are not included. Commonly, index.html is downloaded.

We measure the above four parts using Keynote's [3] tools. Keynote has more than 70 agents in the U.S. and also in some foreign countries, from where we can perform several experiments. They have tools to measure the above four time segments to download Web pages. We use agents inside the U.S. (Washington DC 2, San Francisco 3, Houston 1) and outside the U.S. (Paris 1, London 1 and Tokyo 1). The name signifies the geographical position of the agents. We use four categories of URLs: 100 sites prepared by PC magazine [5], 80 Ph.D. schools in the U.S., 60 URLs in Brazil, and 50 URLs in South Africa. We conduct the experiment around noon (12-2 PM EST), evening (6-8 PM EST) and night (12-3 AM EST). The daytime experiment gives us results when traffic is high inside the US and the nighttime experiment gives us results when traffic is low inside the US. We repeat it for three different days, and for each time we take

three replicas for each URL. The first one gives us the DNS query time that is often not a local copy (Cache Miss). The second and third one normally gives us the cache access time for a DNS query (Cache Hits).

3.1 Bottleneck in DNS Query

Figure 2 (Cache Miss) shows the average time distribution of DNS time, connection establishment time, time to get the first byte, and downloading time for top 100 sites when the measuring agent is in Paris. It takes 0.66 sec to download a 1KB page on average, where 33% is for DNS, 23% for connection setup, 37% to get the first byte after the connection is established, and 7% to download the file. The overall result of our experiment is DNS time 10%-25%, connection setup time 20%-30%, time to get the first byte 40%-60%, and downloading time 10%-20%. The DNS time is significant in accessing international Web sites. Figure 3 shows the same statistics as a replica of Figure 2. Figure 3 shows that cache hits save 90%-98% of DNS

Distribution of time of top access URLs (Cache miss)
Total Time=0.66 sec

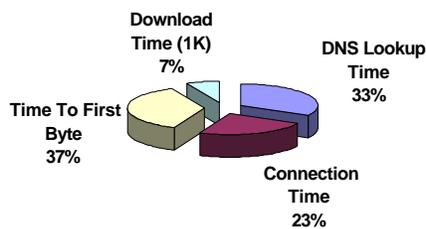


Figure 2: Top 100 URLs from Paris 1 (Cache miss)

Distribution of time of top access URLs (Cache hits)
Total Time= 0.485 sec

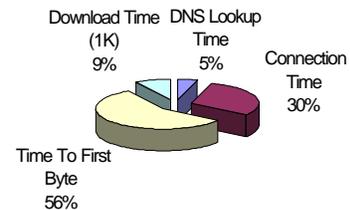


Figure 3: Top 100 URLs from Paris 1 (Cache hits)

An Outlier for top 100 URLs

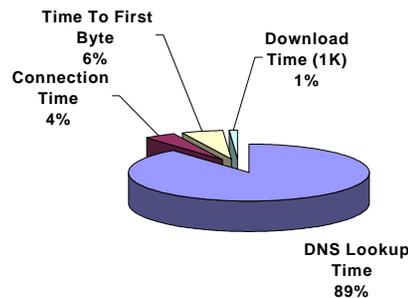


Figure 4: An outlier for top 100 URLs

time. We can mirror the Name Servers, especially root and top-level domains, and distribute them based on geographic location to reduce DNS query time. Figure 4 shows an outlier for Figure 2. We consider those data as outlier where any of the components is exceptionally high from rest of the result.

3.2 Bottleneck in Server

The most interesting fact of this experiment is that around 40%-60% of total latency is taken to get the first byte after the connection is established. A Bellcore Web page [1] observes a similar fraction and attributes it to server delay. According to Barford [2], “This delay is due to the need for the request from the client to get up to user space and then for the response to get back down to the network. This delay is very low when the server is lightly loaded.” We tried to validate their hypothesis but got different results. We used the Keynote agents in Washington DC 2, Paris 1, and Tokyo 1 to access timer.cs.vt.edu (a Pentium II 300 Megahertz machine with 128 MB RAM) running Apache 1.3 on FreeBSD 2.2.7. We used file sizes of 1.58 KB, 7.6 KB, and 20.6 KB. We took several replicas and all showed that the server delay was 1-3 ms.

We imposed an artificial load to the server by using *Webjamma* [7], an artificial HTTP traffic generator, to send a series of URLs to timer.cs.vt.edu. We increased the number of clients such that the server can process 30 requests per sec. We also increased the load as high as 100% (measured using FreeBSD *uptime* for 1 min) by running a simple floating point calculation program in the background to consume CPU cycles. But the server response time did not increase to more than 3 ms. In one case, we got a 19 ms server response time when the server was 25% loaded. We think this is an exception because when it was 59% loaded the server response time was no more than 3 ms. In two other cases, the server sent an ack to the client’s request before sending the actual data. In these two cases, the server response time was around 200 ms. But this 200 ms is insignificant with respect to the corresponding time of getting the first byte, which was 1 sec and 3.58 sec respectively.

We can explain the time taken to get the first byte as a round trip time from the client to the server. The client sends an ack of the server’s SYN packet (connection establishment) and waits for the first packet. So, the client needs to wait for a whole round trip time and for the server’s delay. Because the server delay is very small (1-3 ms), the round trip time dominates. Most of

the time it is pretty close to the connection setup time (CST), which is desirable. But sometimes this time goes high, which makes the overall time to get first byte high. For Figure 2, there are 19 cases when CST is greater than time to get the first byte (GFB) and 27 cases when GFB is 50% more than the CST. The rest of the time, they are pretty close (GFB is more than CST but not more than 50% of CST).

Persistent-HTTP can save the connection time for downloading a page that has more than one embedded image. In our experiment, we didn't use persistent-HTTP because we are mainly concerned about GFB not CST.

We validated the time to get the first byte as a RTT by *ping*. For this purpose, we ran *ping* while the server experiment was going on. In most of the cases, time to get the first byte is close to the average *ping* RTT and sometimes it is as high as the maximum RTT given by the *ping*.

4 Bottleneck in Link or Router

From section 3, we see that Web page latency is dominated by RTT. The question is what makes the RTT high? Link and router characteristics have great impact on RTT. So, let's look at the RTT analysis for links inside the U.S., from the U.S. to foreign countries and vice versa. This analysis will help to explain whether link or router is a bottleneck or not.

The same URLs mentioned before (top 100 sites, top U.S. universities, Brazil and South Africa) are used in this experiment. In each experimental trial, paths from a host connected to domain vt.edu by switched 10 Mbit/sec Ethernet to the remote servers in the URLs are identified using *pathchar* [4]. We also analyze traffic from Bangladesh and South Africa to the United States and used samples of the top 100 sites prepared by PC Magazine [5] for this purpose.

4.1 RTT Analysis (inside the US)

To analyze RTT inside the US, we trace path for URLs of top 80 universities and some of the URLs from top 100 sites prepared by PC magazine [5].

The RTT is order of 20-35 ms in case of East Coast, order of 40-55ms in Central part, and order of 70-90ms in case of West Coast from Virginia Tech. We get some exceptions for example the

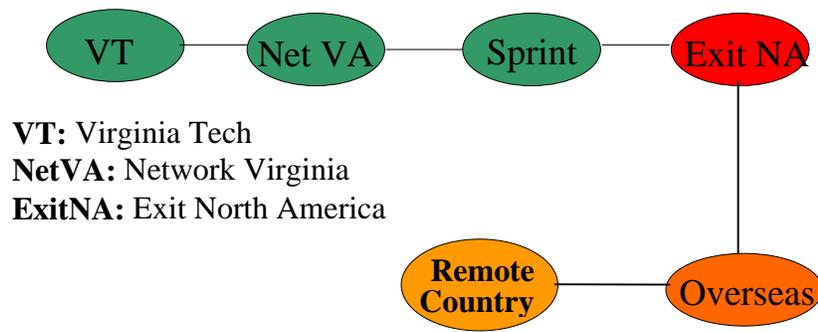


Figure 5: Segmentation of an entire path from U.S. to International countries.

RTT is 213 ms in case of Syracuse University and 118 ms for University of Southwestern Louisiana. Most of the RTT consumed in case of Syracuse University is inside the university.

In the east part, the RTT is almost distributed all over the hops. None of the link/router dominates the overall RTT. But when the traffic goes to the West Coast, either the bottleneck is in Virginia Tech to vBNS (Very high speed Backbone Network Service) connection or in inside Sprint. So, this link or the router is a possible candidate for bottleneck when traffic goes from VT to the West Coast of the US.

4.2 RTT Analysis (the US to foreign countries)

An entire path from source to destination machine is divided into several segments, see Figure 5. Normally the segmentation is done based on network domains. Each segment may have one or more hops such as VT has 3 hops, the Overseas has one hop but Sprint has different number of hops in different experiments. Please see Habib [8] for details. So, a packet may follow different routes within each segment.

Figure 6 shows the mean, maximum, with mean+stdev, and mean-stdev of RTT experienced in different segments of Figure 5 when traffic goes to foreign country from the U.S. A very high RTT is experienced inside Canada and the U.S., specifically in the ExitNA segment. ExitNA segment has 40 times more RTT than VT (LAN). The standard deviation of RTT in this segment is very low, which means the segment often has this high RTT. We also observe that the Teleglobe network shows unusual behavior in the sense that it has high bandwidth, but also a high RTT. The question is why ExitNA segments exhibit a very long RTT. For this purpose, we identified the links that account for most of the long RTT. We call these links *busy links*. Table 1

RTT through Alter.Net (SA)

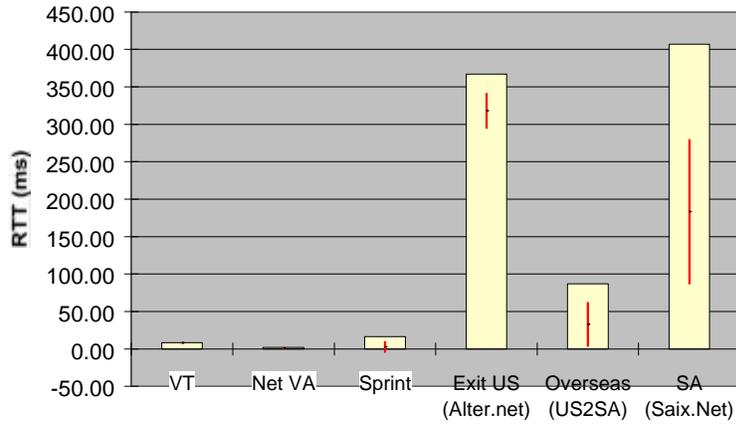


Figure 6: RTT from the U.S. to South Africa. The red line connects all mean values. It also shows mean+stdev, mean-stdev and max(bar height)

shows the end to end routers that are connected with the busy link, the bandwidth of the links and their RTT.

Table 1. Routers that connect busy links, their BW and RTT

Countries	Links (Router to Router)	Link BW	Link RTT
United States to Brazil	gip-dc-3-fddi0-0.gip.net (204.59.144.197) to 204.59.224.202	1.5 Mb/s	140 ms
	gip-penn-3-fddi0-0.gip.net (204.59.136.197) to 204.59.192.22 / 204.59.192.100	1.5 Mb/s	145 ms
	gip-ftworth-2-fddi1-0.gip.net (204.59.120.194) to 204.59.121.14	1.5 Mb/s	150 ms
	gin-mtt-bb2.Teleglobe.net(207.45.223.61) to embratel-gw4.Teleglobe.net(207.45.206.178)	20~70 Mb/s	315 ms
United States to South Africa	195.ATM-0-0.Gw1.DCA1.ALTER.NET (146.188.161.17) to telkom-gw.customer.ALTER.NET(137.39.128.14)	483 Kb/s	325 ms
	Fddi0-0.New-York4.NY.Alter.Net(137.39.126.10) to satelkom-gw.customer.ALTER.NET(137.39.245.238)	1~4 Mb/s	300-325 ms
	gip-dc-3-fddi0-0.gip.net (204.59.144.197) to 204.59.225.86	1 Mb/s	250 ms
	gip-dc-2-fddi1-0.gip.net (204.59.144.194) to 204.59.145.14	1 Mb/s	240 ms

There are some common properties of the busy links such that each link is in Canada or inside the U.S. and entirely in one network domain. The end routers connect the busy link are in very close considering physical proximity, so they are not connected by a satellite link. In most of the cases the first of the two routers uses FDDI and the link is the final network link before traffic

leaves the U.S. or Canada to the overseas link. Based on the data, graphs and discussion above, the causes of high RTT can be hypothesized as:

A busy link router either has insufficient processing capacity or insufficient outgoing link capacity.

To validate our hypothesis we, measure the queuing delay throughout the path. Table 2 shows the total queuing delay at routers along the paths to South Africa. Here, all U.S. segments except the busy link are considered a single segment named “U.S. Link.”

Table 2. Queuing delay for each segment from the U.S. to South Africa

	Min	Median	Std	Mean	Max
US Link	1.00	5.83	47.83	21.75	163.58
Busy Link	0.00	42.60	24.58	36.96	61.20
Overseas	0.00	23.80	29.88	30.54	80.00
SA	4.48	64.75	209.29	148.68	635.00

It shows that the queuing delay for routers within the remote country is an order of magnitude higher than the rest of the path. This is explainable because the network in the remote countries has smaller bandwidth. But what is more interesting is that the queuing delay at the *single router*, at the entrance to the busy link, is more than the *entire* queuing delay at all routers within either the U.S. Link segment or the Overseas segment. Thus improving this single router is critical to improving the RTT up to the entry of packets into the remote country.

4.3 RTT Analysis (Foreign countries to the US)

Figure 7 shows the RTT for the path from South Africa (SA) to the United States. Here, we divide the path in three segments: inside SA, exit SA and inside the US. The bottleneck bandwidth is very low (127Kbps) in South Africa. It introduces significant amount of delay like 100ms. But the main delay is introduced by the link exiting SA and it is more than 500ms. Traffic analysis from Bangladesh to the US shows that a very high latency is imposed by satellite connection (order of 750 ms) as well as the transoceanic link (order of 150 ms). So, we can summarize, the cause of high RTT for a incoming packet to the U.S. is as follows:

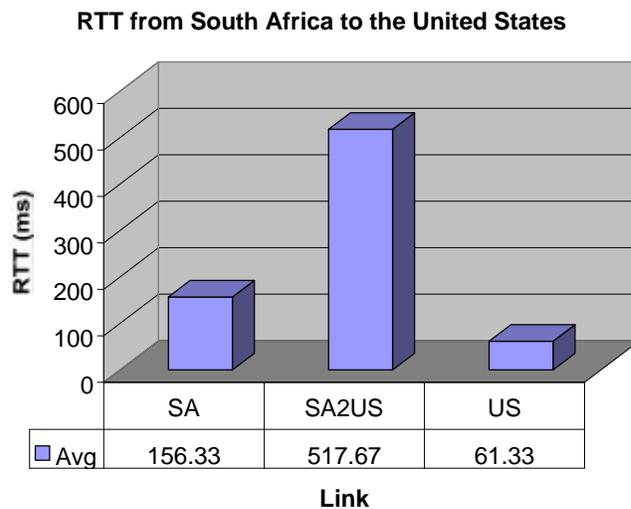


Figure 7: RTT from South Africa to the U.S.

- Satellite connection anywhere in the path causes high delay
- Link connecting two countries often experience high traffic and causes high RTT

4.4 Summary

From two-way traffic analysis, we see that for outgoing traffic the RTT bottleneck is in the US and just before leaving the country. Incoming traffic to the US does not subject to high latency after its entering. So, we conclude that for an international route a router (e.g. Table 1) is a strong candidate for bottleneck. This behavior also exhibits inside the US (RTT is mainly consumed in the transition link from East Coast to West Coast).

5 Bottleneck in TCP

TCP uses 3-way handshake to establish a connection and 4-way handshake to release a connection. So, to download a small size file, the overhead is very high. TCP slow start is also causes overhead in file transferring and especially for small file. Again if TCP needs several unnecessary retransmissions due to incorrect guess of its timer value, then overall downloading time will be very high. We want to investigate whether or not we can use TCP more efficiently. So, our research question is how often TCP generates unnecessary retransmissions and how many of them can be saved. We conducted experiments for FTP GET, FTP PUT and HTTP GET, for four different sites (inside as well as outside the U.S.), for 3 different file types and sizes, for 3 different times of a day, and for two different days of a week. We used *tcpdump* [6]

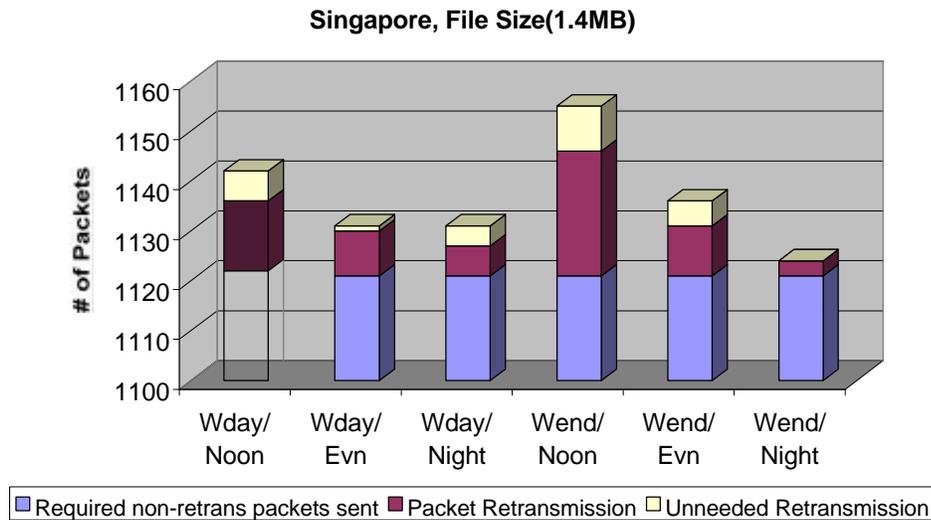


Figure 8: FTP operation from the U.S. to Singapore for a 1.4MB file

to trace packets at transport layer level. To get the details version of this experiment please see Habib [9].

We got the result that unnecessary retransmission mainly happens for large file. Figure 8 shows how many packets are required to send the file without retransmission, how many retransmitted packets were sent, how many of retransmitted packets were judged to be unnecessary to transfer a large file, we use 1.4 MB, using FTP. In this case 50% to 60% of retransmission could be saved. For a Medium file, we use 490 KB, transfer with potential savings in retransmissions is 22% to 83%. Unlike the Large and Medium file sizes, small file, we use 20 KB, shows almost no retransmission. For small file, unnecessary retransmission rarely happens even in international links.

An interesting result is that in LAN, TCP causes many unnecessary retransmissions. TCP performs well inside the US and causes many unnecessary retransmissions for the international countries. We can summarize the worst case percentage of retransmission and unnecessary retransmission with respect to the required number of packets sent in Table 3 (The worst case is the largest number for the six times of day for each case). From the table at most 7% of the packets are retransmitted, and no more than 3.4% of the retransmissions are unnecessary.

Table 3: Worst case percentage of retransmission and unnecessary retransmission. Percentage is with respect to required packets sent.

File (No of Packets)	VT (Retransmission)		USC (Retransmission)		South Africa (Retransmission)		Singapore (Retransmission)	
	Total	Unnecessary	Total	Unnecessary	Total	Unnecessary	Total	Unnecessary
Small (14)	0	0	0	0	7.14%	0	7.14%	0
Medium(343)	4.06%	1.45%	2.45%	0	5.52%	3.40%	2.45%	0.54%
Large (1050)	5.32%	2.28%	0.89%	0	4.19%	0.38%	2.20%	0.8%

We can explain the causes of unnecessary retransmission that are observed in this TCP experiment as follows:

First, when transferring a large file, TCP will increase the congestion window so that a large number of packets are sent back-to-back. If another sender on another TCP connection starts transmitting, it may cause losses in the first connection until that first connection reduces its congestion window. This phenomena does not occur for the small file case, because TCP's slow start algorithm starts the congestion window at one packet, and slowly increases the size.

A second reason is that most of these retransmissions are due to creation of holes in the sequence number space. This hole was created for a series of packets that got lost or came out of order. So, when the sender resends the missing series of packets, some of the accepted packets are also resent this time, causing the unnecessary retransmission. This problem would be solved with the selective acknowledgement [10] option for TCP.

A third reason is that TCP may incorrectly guess the timeout value for its timer due to high variance of RTT. This problem can be overcome by fine-tuning the formula $RTO = RTT + 4 RTT_{var}$ [10, 18]. Increasing RTO won't hamper the performance because we have useful mechanism like fast retransmit [10].

6 Conclusion

We investigate all the possible sources of latency. Reducing DNS time will save overall downloading time. To reduce DNS time, the number of cached entries can be increased in local DNS server. We can also force to cache entries for popular servers. The next source is

connection establishment time. If we use persistent-HTTP, then connection time for more than one file can be saved. High RTT causes connection time, time to get the first byte, and downloading time high. The RTT depends on Link and Router. Link bandwidth is increasing day by day. Our study shows that router is a great source of bottleneck both for inside the US and international countries. To increase the performance of router, we need more investigation. TCP's timeout calculation can be fine-tuned again to avoid unnecessary retransmission. A summary is shown in Table 4.

Table 4: Sources of bottleneck and how to overcome them

Source	Bottleneck	How to Overcome
DNS Query	Yes, for international sites	1. Increase cached entries in local DNS server. 2. Force to cache entries for popular servers.
Server	No	
Link	Yes, especially in transition links	1. Increase bandwidth will definitely help. 2. More transition links to the foreign countries.
Router	Yes	1. Investigate Router parameters like buffer, computing capability, etc. 2. More gateway to international countries
TCP	Yes	1. Overcoming Handshaking problem in connection establishment and release, 2. Get rid of Slow-start 3. Fine-tuning the TCP timer value.

Acknowledgements

Thanks go to Eric Siegel for giving us the facility to use Keynote tools. We also want to thank Hussein Suleman, Sabbir Ahmed, Moniruzzaman Mozumder, and Abdul Maleq Khan for giving us permission to use their machines. IntelSat and NSF grants CDA-9312611 and NCR-9627922 partially supported this work.

References

- [1] Christian Huitema, *Internet quality of service assessment*,
ftp://ftp.bellcore.com/pub/huitema/stats/quality_today.html

- [2] Paul Barford and Mark Crovella, "Measuring Web Performance in the Wide Area", Computer Science Department, Boston University. Submitted to performance review. March 1999 <http://www.cs.bu.edu/students/grads/barford/Home.html>
- [3] Keynote Systems Inc., <http://www.keynote.com>, 1998
- [4] Jacobson, V., C. *Pathchar*, <ftp://ftp.ee.lbl.gov/pathchar.tar.Z>
- [5] Top 100 sites from PC Magazine <http://www.zdnet.com/pcmag/special/web100/index.html>
- [6] Jacobson V., Leres C., and McCanne S., *tcpdump*, available at <ftp://ftp.ee.lbl.gov/tcpdump.tar.Z>.
- [7] *Webjamma*, WWW Traffic Analysis Tool, <http://www.cs.vt.edu/~chitra/webjamma.html>
- [8] M. Ahsan Habib, Marc Abrams, *Analysis of Bottlenecks in International Internet Links*, Technical Report TR-98-24, Computer Science Dept., Virginia Tech, Dec 1998
- [9] Md. Ahsan Habib, Marc Abrams, *Analysis of Unnecessary Retransmissions in TCP*, Technical Report TR-99-3, Department of Computer Science, Virginia Tech, April 1999
- [10] W. Richard Stevens, "TCP/IP Illustrated", Vol. 1, Addison Wesley, 1998
- [11] Ghaleb Abdulla, Edward A. Fox, Marc Abrams, "Shared User Behavior on the World Wide Web," *WebNet97*, Toronto, October 1997. <http://www.cs.vt.edu/~nrg/pubs.html>
- [12] Vern Paxson, "End-to-end routing behavior in the Internet," *In Proceedings of ACM SIGCOMM '96*, Palo Alto, CA, August 1996.
- [13] Apache HTTP Server Project, <http://www.apache.org>, 1998.
- [14] Mogul, J. "The Case for Persistent-Connection HTTP," *Western Research Laboratory Research Report 95/4*, <http://www.research.digital.com/wrl/publications/abstracts/95.4.html>, Digital Equipment Corporation, May 1995.

- [15] Padmanabhan, V. N. and J. Mogul, "Improving HTTP Latency," *Computer Networks and ISDN Systems*, v.28, pp. 25-35, Dec. 1995. Slightly Revised Version in *Proceedings of the 2nd International WWW Conference '94: Mosaic and the Web*, Oct. 1994.
- [16] Touch, J., J. Heidemann, K. Obraczka, "Analysis of HTTP Performance," USC/Information Sciences Institute, June, 1996.
- [17] Spero, S., "Analysis of HTTP Performance Problems," <http://www.w3.org/Protocols/HTTP/1.0/HTTPPerformance.html>, July 1994.
- [18] Jacobson, Van, "Congestion Avoidance and Control," *Proceedings of ACM SIGCOMM '88*, page 314-329, Stanford, CA, August 1988.
- [19] Heidemann, J., "Performance Interactions Between P-HTTP and TCP Implementation," *ACM Computer Communication Review*, 27 2, 65-73, April 1997.
- [20] Thompson K., Miller G. J., Wilder R., "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network*, Nov/Dec 1997
- [21] Bernardo A. Huberman, Peter L.T. Pirolli, James E. Pitkow, and Rajan M. Lukose, "Strong Regularities in World Wide Web Surfing," *Science* April 1998