Measuring the Web Using a Versatile Meta Information Crawler

Ting Liu Andrea LaPaugh Larry Peterson

Abstract

In this paper, we present data which characterizes three aspects of Web interactions: failures, timing performance, and protocol compliance. We collected the data using our Versatile Meta Information Crawler, which is designed to acquire a wide sample of the Web, accurately recording its behavior and performance, and building a large repository of Web page meta information. We have crawled 300,000 Web pages under 130,000 domain names and 90,000 IP addresses that are dispersed throughout the Web. The major findings are as follows. For failures, the likelihood of encountering a Web failure is 12%. DNS failures account for 50% of all the communication failures, and "URL Not Found"s account for 90% of all the transaction failures. For timing performance, none of the communication phases dominates the entire Web transaction. We examine each phase in more detail to identify its empirical parameters. For protocol compliance, persistent connections are not indicated properly by major servers, and conditional GET is not sufficiently supported. Based on the data, we suggest a number of system improvements.

1 Introduction

With the rapid growth of the World Wide Web, measuring and characterizing its behavior and performance has become essential to understand its nature and find implications for its improvements. However, two fundamental properties of the Web make this challenging. The Web is dynamic; therefore actual measurements are needed to accurately capture its live activities. The Web is made up of a huge number of heterogeneous sites; therefore the measurements must be extensive to reflect its entirety.

In this paper, we present a new approach to study the Web using our Versatile Meta Information Crawler (abbreviated as VerMIC). This approach is aimed at accessing a wide sample of the Web, accurately recording its behavior and performance, and building a large repository of Web page meta information. We consider VerMIC to be a powerful research tool because of its extensive coverage, use of actual measurement versus simulation, fined-grained and user-oriented measurement, and flexible functionality.

Throughout our study, we seek to answer the following questions:

- What failures are encountered on the Web, and how prevalent are they?
- What performance does the Web deliver, and what has impact on this performance?
- How well are Web protocols complied with, and what are the problems?

Our study yields the following main results:

- We characterize the prevalence of Web failures and find that human-introduced errors are the most damaging to the success of Web transactions.
- We evaluate the performance of static pages and dynamic pages, and identify the empirical parameters of each communication phase.
- We examine the status of protocol compliance, and point out where improvements can be made in the use of protocol performance features.

The rest of this paper is organized as follows. Section 2 describes the methodology of our measurements. Section 3, 4, and 5 present the detailed statistical results for Web failures, performance, and protocol compliance. These results suggest a number of promising system policies and enhancements such as DNS and URL negative caching, conservative use of redirections, parallelization of dynamic page construction and transmission, discreet use of TCP slow start, proper indication of persistent connections and sufficient support for conditional GET. Our sugggestions are discussed in detail in Section 6. Section 7 compares our work with previous efforts. And we conclude in Section 8.

2 Versatile Meta Information Crawler

This section describes the design of our meta information crawler VerMIC in terms of its unique model and strategies. We believe VerMIC is a powerful research tool because it has the following instrumental properties.

First, the coverage of VerMIC is extensive and scalable not only in terms of the volume of pages, but also in terms of the number of administrative and geographical domains of the Internet. This ensures our data is representative of the entire Web.

Second, VerMIC uses actual measurements, not simulations, to produce the most accurate characterization. It gives on-the-spot reports and allows empirical modelling and the identification of real-time real-world problems of the Internet. This information can aid studies of pratical issues and facilitate the design of simulation-based systems.

Third, the measurements of VerMIC are fine-grained and user-oriented. Its measurements examine individual communication phases of each Web transaction and every single event of information delivery. Since measurements are initiated as the activity of a typical end-user, they can illustrate the current Web status as other end-users should see it.

Finally, the functionality of VerMIC is flexible. It does not rely on any existing information resource and can take on various new features with little effort.

2.1 Model Used by VerMIC

In contrast with conventional Web crawlers whose sole target is the documents on the Web, VerMIC measures the aspects of Web transactions. It regards a Web transaction as consisting of five communication phases: the DNS Phase, the Connection Phase, the Request Processing Phase, the Response Transmission Phase, and the Redirection Phase if applicable.

In the DNS Phase, the Web client tries to find out the IP address of the Web site by engaging one or more name servers to translate the domain name into an IP address. The DNS request will either be resolved directly in the cache of the local name server, or it will go to the root name server, a second level name server, and so on, until a match of the whole name is returned. After the domain name translation, the client will try to establish a TCP connection to the Web server by a three-way handshake in the Connection Phase. Once the connection has been set up, the client will send an HTTP request to the server, who will in turn process the request and generate a corresponding HTTP message as the response. This is called the Request Processing Phase. In the last step, the Response Transmission Phase, the response will be sent over the network to the client in a series of TCP packets. Under certain circumstances, an HTTP request can be redirected by the server to a different URL. If a request is redirected multiple times, a single Web transaction will involve a sequence of hops. We use the Redirection Phase to represent the communication for all the intermediate hops, and the prior four phases refer to the last hop only, although in practice they are repeated on each hop.

Every time a Web transaction is completed, VerMIC generates an exhaustive report that contains three transaction records of interest.

2.1.1 Failure Record

The failure record contains any perceived failure and related diagnostic information. We differentiate between communication failures and transaction failures. Any failure will prevent the Web transaction from proceeding to completion.

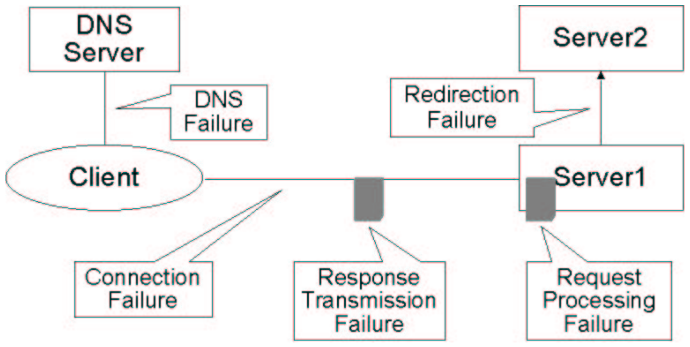

Figure 1: Communication Failures

Figure 1 summarizes all the communication failures that VerMIC has encountered, They are described below.

DNS Failure In the DNS Phase, the domain name translation may fail and no IP address is returned. This is indicated by an error returned from the name lookup.

Connection Failure In the Connection Phase, the three-way handshake may fail and no connection is established. This can be detected either immediately from the feedback of the connecting socket or after a long waiting period. We define a wait-based connection failure by a connection time-out of 30 seconds, which is justified by the connection time distribution shown in Section 4.

Request Processing Failure In the Request Processing Phase, the server may fail to process the request and generate no response. This can be detected either immediately from a cut off of the connection or after a long waiting period. In the second case, we set a request processing time-out to be 150 seconds to prevent infinite waiting. This limit is also justified by the relevant timing statistics.

Response Transmission Failure In the Response Transmission Phase, the transmission may fail, and only a partial response may be transmitted. This can be detected in the same way as the Request Processing Failure. We set a response transmission time-out to be 30 seconds, which is the largest interval between any pair of successive packets.

Redirection Failure In the Redirection Phase, a server may fail to resolve the next hop so that the next request cannot be issued. In some cases, the server does not provide any URL in a redirection message. In other cases, the request has already gone through many hops, even looping, and further redirection seems unpromising. We set the maximum number of hops a Web transaction should experience to be 4, and consider any additional attempted hops to be a redirection failure.

Status Code	Definition
404	URL Not Found
403	Forbidden (URL Not Accessible)
401	Unauthorized User
500	Internal Server Error
400	Malformed Request
Other	N/A

Table 1: Transaction Failures

Table 1 enumerates the major transaction failures. In contrast with the communication failures, the client is notified of these failures through a successfully transmitted response, and no more communication between the client and the server is needed. The status code in the final response directly indicates why the transaction has failed.

In Section 3, we will present the detailed statistical results of all the Web failures. The collection of the failure records will help us identify which parts of the system can be at fault and how frequently they are, and give indications of what measures should be taken to reduce their effects.

2.1.2 Timing Record

The timing record contains the timing measurements of the five communication phases. We define the durations of the five phases to be the Redirection Time, the DNS Time, the Connection Time, the Request Processing Time, and the Response Transmission Time, respectively. And we defined the duration of a Web transaction to be their sum.

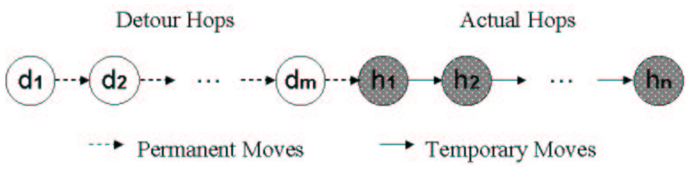

Figure 2: Actual Hops and Detour Hops

Rediretion Time A redirection message could indicate a permanent URL move, meaning that future requests should be directly sent to the new URL, or a temporary URL move, meaning that future requests should still be sent to the original URL. In a sequence of redirection moves, once a temporary move is encountered, all the subsequent redirection hops of either type will be part of future accesses as well. Therefore the hop that issues the first temporary move, marked as h_1 in Figure 2, should be the actual start point of the Web transaction. Any subsequent permanent move should be considered as temporary as well. We split the hops into actual hops that are after and including h_1 and detour hops that are before it. Only the actual hops except the last one participate in the calculation of the redirection time.

$$redirection\ time = \begin{cases} \sum_{i=1}^{n-1} time_{h_i} & \text{if} \quad n > 1; \\ 0 & \text{if} \quad n = 1. \end{cases}$$

where $time_{h_i}$ is the time spend on the ith actual hop. The next four times refer to the last actual hop h_n only.

DNS Time To the first approximation, we define the DNS time to be the time for name translation on h_n. However, if the same name has occurred in the detour path and has been cached in the local name server, the DNS time on h_n will be much shorter than if the detour path is skipped. Therefore, we adopt a calibration.

```
if ($host\ name_{d_i} == host\ name_{h_n}$) {
  $DNS\ time = DNS\ time_{d_i}$;
}
```

where d_i is the first detour hop that has the same host name as h_n.

Connection Time We define the connection time to be the time to establish the connection on h_n.

Request Processing Time We define the request processing time to be the interval between when the request to h_n is sent and when the first response packet from it is received.

Response Transmission Time We define the response transmission time to be the time to transmit all the rest of the packets, if any. We also record the size and arrival time of each packet so that the transmission pattern and rhythm is completely documented.

In Section 4, we will present the detailed statistical results of all the performance measurements. The collection of the timing records will help us identify the empirical parameters of all the individual communication phases and their interrelationships. This has implications for performance improvements.

2.1.3 Protocol Record

At the time of writing, two versions of the HTTP protocol are prevalent. HTTP/1.1 was recently standardized and is young in terms of development and adoption. HTTP/1.0 has existed for a long time and is still influential.

The protocol record summarizes the protocol behavior of the server in three aspects. The first aspect is whether the server adheres to the syntactic requirements of the protocol. The second one questions whether the server exploits the performance features that the protocol offers. The third aspect examines how many miscellaneous communication features are relevant to the server.

Syntactic Requirements Both HTTP/1.1 and HTTP/1.0 have syntactic requirements that reinforce the format of their messages. Incorrect syntax will directly lead to misinterpretation of the message or more implementation efforts to accommodate its malformation.

Performance Features Both HTTP/1.1 and HTTP/1.0 offer a number of features that are aimed at improving the protocol performance. Persistent connections introduced in HTTP/1.1 allow a client to make multiple requests without waiting for each response. This technique is called pipelining in HTTP and can be used to avoid roundtrip delays and reduce the number of packets. Conditional GET, introduced in HTTP/1.0 and expanded in HTTP/1.1, allows conditional transmission of the requested document. It is used by proxy servers in cache validation to reduce the volume of transmissions when the cached entry is up to date and avoid extra roundtrips if it is stale. How widely these features are supported directly determines how often performance enhancements can be obtained.

Communication Features Both HTTP/1.1 and HTTP/1.0 provide many other miscellaneous features that are aimed at improving client-server communication. How many HTTP/1.1 innovations are adopted and how many HTTP/1.0 legacies are present in HTTP/1.1 is a good indication of which features are desirable and should be inherited and further developed.

In Section 5, we will present the detailed statistical results of all the aspects of protocol behavior. The collection of the protocol records will help us understand how protocol features are used, and give guidance for the evolution of protocols.

2.2 Strategies of VerMIC

VerMIC has a crawling strategy and a storage strategy. The crawling strategy decides the order of the URLs to crawl. Our goal is to crawl pages that are dispersed throughout the Web so that our data is representative of the entire Web. We are not interested in which URLs lead to the most popular pages, as a conventional Web crawler might be. Therefore, breadth-first is our primary crawling strategy. This will open up as many Web territories as possible.

To further distribute visited URLs, we cache the recently visited domain names, and defer the next URL if its domain name is semantically related to any of the cached ones. We tell the similarity between domain names using prefix and suffix matching. For example, there is similarity between *www.yahoo.com* and *music.yahoo.com*, and between *www.ebay.com* and *www.ebaycareers.com*. This method also prevents our crawler from generating an overwhelming workload on a single server.

As the start point of the crawl, we select the 300 most popular Web sites from *www.100hot.com*. These include 100 *.com*s, 100 *.edu*s, *.org*s, and *.net*s, and 100 Web sites whose domain names are outside the United States. We begin with popular sites rather than random sites because they tend to have abundant links to other sites that have little affiliation with the original sites, and hence are more suitable to direct the crawl.

The storage strategy defines the organization of the crawled data. One of our important decisions is to distinguish static pages and dynamic pages. Static pages are retrieved from the server's disk and passed to the client without changes. Dynamic pages are generated by a program run on the server when the page is requested and can be different each time requested. Their different service patterns suggest different performance details, which are verified in Section 4.

Five-day Experiment	Static	Dynamic
Pages Crawled	240,654	53,758
Domain Names Accessed	128,121	19,725
IP Addresses Accessed	87,048	16,618

Table 2: Data Volume of Five-day Experiment

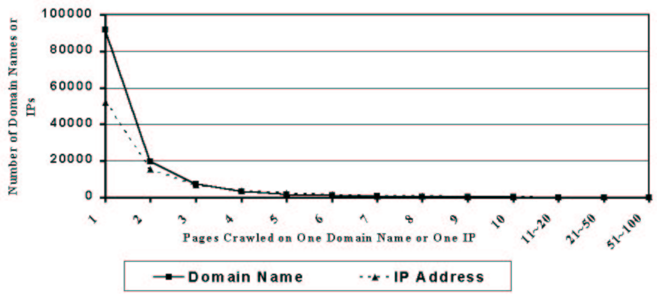

Figure 3: Distribution of Static Pages across Domain Names and IP Addresses

Table 2 shows the data volume of the five-day experiment starting from November 23, 2001. To demonstrate the effectiveness of our extensible crawling strategy, Figure 3 shows the distribution of static pages across domain names and IP addresses. Figure 4 shows the partition of major domains in crawled pages.

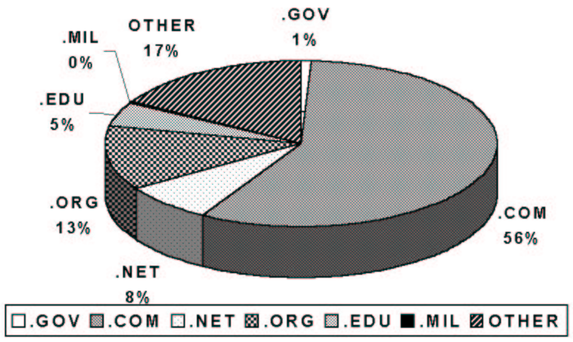

Figure 4: Partition of Major Domains

3 Web Failures

Based on the failure records of all the Web transactions, we have the following results.

(1) The likelihood of encountering a Web failure is about 12%
(2) The chances for communication failures and transaction failures are approximately the same
(3) DNS failures account for almost 50% of all the communication failures
(4) "URL Not Found"s account for almost 90% of all the transaction failures
(5) Human-introduced errors are the most damaging to the success Web transactions

In the rest of this section, we will present our analysis only on static pages, because cases are similar for dynamic pages.

3.1 Prevalence of Web Failures

	Count	%
Successful Transactions	240,654	85.9%
Communication Failure	14990	5.4%
Transaction Failures	17678	6.3%
Beyond Experimental Constraints	6696	2.4%

Table 3: Prevalence of Web Failures

Table 3 shows the prevalence of Web failures. The likelihood of encountering a Web failure is about 12%, and the chances for communication failures and transaction failures are approximately the same. Huge documents and those protected by the robot exclusion protocol are listed as "beyond experimental constraints".

3.2 Communication Failures

Table 4 shows the occurences of all the communication failures, of which DNS failures account for almost 50%.

Failure Type	Count	%
DNS Failures	7397	49.3%
Connection Failures	5143	34.3%
Redirection Failures	1517	10.1%
Response Transmission Failures	486	3.2%
Request Processing Failures	447	3.0%

Table 4: Communication Failures

3.2.1 DNS Failures

We consider two possible causes for DNS failures. One is that the domain name is invalid, and the other is that there's an internal error in the name server hierachy. Since in practice, each zone in the domain hierarchy is implemented in two or more name servers for the sake of redundancy, we speculate that a system error is much less likely than a bad name. To verify this, we re-crawled all the 7397 pages that had DNS failures, and only 43 of them recovered in the second attempt.

All names, including invalid ones, were obtained from crawled pages. The invalid names either had a typing error when created or became invalid afterwards. We conclude the main source of DNS failures is insufficient human maintenance on Web pages. Also, since following the links in one page to browse other pages is the common behavior of Web users, we expect that the number of occurences of DNS failures encountered by typical Web users is at the same level as our experience.

We find that the average time to return a DNS failure is 2 seconds.

3.2.2 Connection Failures

Diagnostic	Count
Connection Time-out	3988
Connection Refused	1054
Routing Problem	93
Other	8

Table 5: Connection Failures

Table 5 shows the classification of the connection failures. A connection time-out could occur when the remote host is not in place, or is temporarily not responding, or is too busy to accept the connection. To verify which of the three cases is the most probable, we repeatedly probed the servers that had connection time-out. It turned out that 80% of the remote hosts were permanantly gone.

A refused connection could occur when the remote host is alive but there's no server application to accept the incoming connections. Although this has rarely occurred, a routing problem can also block the connection establishment.

These numbers show that the main source of connection failures is human misconfiguration of devices, not the physical limitation of devices.

3.2.3 Redirection Failures

Diagnostic	Count
More Than 4 Hops Attempted	1488
Next Hop Not Provided	29

Table 6: Redirection Failures

Table 6 shows the classification of the redirection failures. The two sub-cases have been described in Section 2.

3.2.4 Response Transmission Failures

Diagnostic	Count
Response Transmission Time-out	449
Connection Closed	37

Table 7: Response Transmission Failures

Table 7 shows the classification of the response transmission failures. A connection could be closed by the server when the server withdraws the transmission before its completion; no reason is given. A response transmission time-out could occur when the network is heavily congested and the transmission is too slow. A heavily congested network can also cause a connection time-out or a request processing time-out if the packet cannot be delivered within a reasonable period of time. However, we believe that its effect is much more evident here. In the connection phase and the request processing phase, only a single packet is transmitted in a roundtrip. However, in the response transmission phase, a burst of packets could be transmitted in a roundtrip and result in high loss rate and long delay.

3.2.5 Request Processing Failures

Diagnostic	Count
Request Processing Time-out	275
Connection Closed	106
Other	66

Table 8: Request Processing Failures

Table 8 shows the classification of the request processing failures. Failures of this type rarely occurred, meaning a successful name translation and a successful connection setup almost insure an active and capable Web server. However, in the case that the server is overloaded and cannot respond promptly, a request processing time-out will be issued. In other cases, we find that the server refuses to process the request and closes the connection with no reason given.

3.3 Transaction Failures

Diagnostic	Count	%
URL Not Found	15807	89.4%
Forbidden (URL Not Accessible)	982	5.6%
Unauthorized User	281	1.6%
Internal Server Error	235	1.3%
Malformed Request	160	0.9%
Other	213	1.2%

Table 9: Transaction Failures

Table 9 shows the occurences of all the transaction failures, of which "URL Not Found"s account for almost 90%.

As the most common types of transaction failures, "URL Not Found"s are caused by invalid URLs, and "Forbidden"s are caused by inaccessible URLs. The invalid and inaccessible URLs were obtained from the crawled pages, as were all other URLs. Therefore, the main source of transaction failures is of human origin, such as inaccurate URL citations and citations to URLs that no longer exist.

4 Performance Measurements

Based on the timing records of all the successful Web transactions, we have the following results.

(1) The redirection time of dynamic pages is three times as much as that of static pages; the request processing time is twice as much; but the whole duration is only 27% longer

(2) None of the communication phases dominates the entire Web transaction

(3) The redirection time grows proportionally with the number of intermediate hops; in the case of three intermediate hops, the average is 66% of the total time

(4) For dynamic pages, the request processing time grows linearly with the page size; while for static pages, the request processing time is almost constant.

(5) The common 5-20KB pages have the worst response transmission throughput due to the effect of TCP slow start

(6) The response transmission of approximately 13% of pages have only a single packet in the first roundtrip, which may cause a 200ms delay of the next roundtrip

4.1 Static Pages vs. Dynamic pages

Figure 5 shows the average timing values of static and dynamic pages. There are two interesting characteristics in

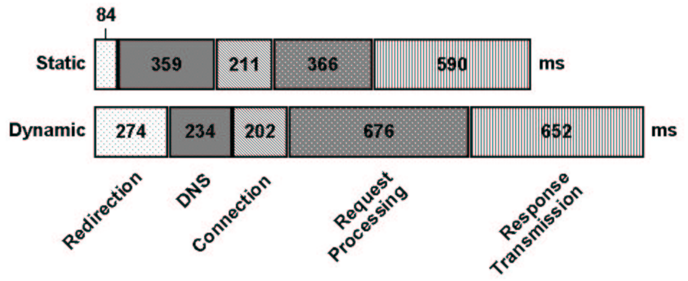

Figure 5: Static Pages vs. Dynamic Pages

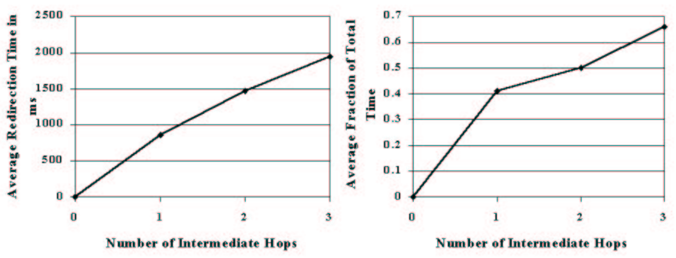

Figure 6: Redirection Time vs. Number of Intermediate Hops

this figure. First, two timing components of dynamic pages are much larger than those of static pages, but the entire Web transaction of dynamic pages is not significantly slower. The average redirection time of dynamic pages is more than three times as much as that of static pages. This is because the server program that processes the dynamic page request often generates local redirections. In some cases, it transforms an external URL to an internal one. In other cases, it hands over this request to another program. Both will cause the client to re-send the request to a new URL. The average request processing time of dynamic pages is almost twice as much as that of static pages. This is because the server program that processes the dynamic page request has to generate the page on the fly instead of retrieving an existing one. However, the entire Web transaction of dynamic pages is only 27% slower than static pages. This is because dynamic DNS time is only two thirds of static DNS time and all other timing components of static pages and dynamic pages are comparable. We have no hypothesis for the difference in DNS times. Second, none of the timing components dominates the entire Web transaction. This is true for both static pages and dynamic pages. For static pages, the response transmission time has the largest share, which is 30% of the total time in average. For dynamic pages, both the request processing time and the response transmission time are the largest, each taking 30% of the total time.

4.2 Redirection Time

Number of Intermediate Hops	%
0	74.1%
1	17.9%
2	5.5%
3	2.4%

Table 10: Multiple Redirection Hops

Table 10 shows the occurences of multiple redirection hops by percentage. Figure 6 shows the relationship between the redirection time and the number of intermediate hops. Both of these are based on dynamic pages. We have similar results for static pages. The redirection time has a proportional growth with the number of intermediate hops, and its percentage of the total time has a linear growth with it. In the case of three intermediate hops, its percentage is as much as 66%, This is surprising, considering that it means two thirds of the transaction time is spent looking for the right URL.

4.3 DNS Time

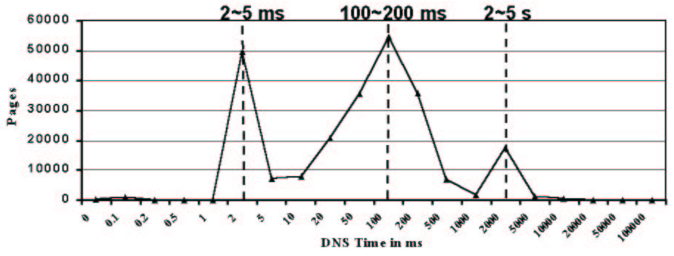

Figure 7: Page Distribution of DNS Time

Figure 7 shows the page distribution of DNS time, based on static pages. We have similar results for dynamic pages. We find three peaks in the curve. We believe these match the three levels of name lookup. Specifically, the first peak represents the DNS requests satisfied by local name servers, the second peak represents the DNS requests forwarded to and resolved by root name servers, and the third peak represents the DNS requests redirected to the second level name servers. If this conjecture is true, we have the following interesting conclusions. First, the difference in the cost of name translation by the different domain name levels is large: increasing by multiplicative factors of 20 to 50. Second, the percentages of the DNS requests resolved respectively by local name servers, root name servers, and the second and even upper-level name servers are 24%, 67%, and 9%.

4.4 Connection Time

Figure 8 shows the page distribution of connection time based on static pages. We have similar results for dynamic

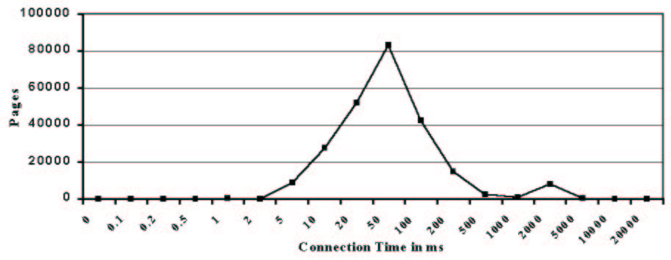

Figure 8: Page Distribution of Connection Time

pages. The flattened part on the right side of the curve justifies that the 30-second connection time-out we set is reasonable.

4.5 Request Processing Time

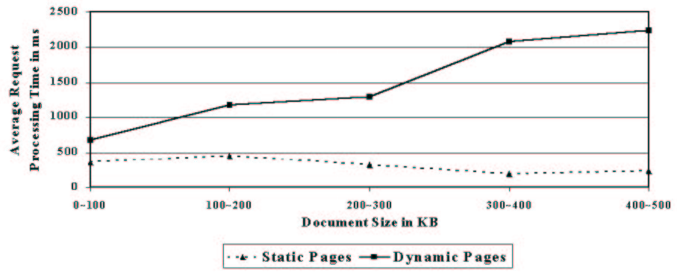

Figure 9: Request Processing Time vs. Document Size

In Figure 9, the flat curve for request processing time vs. document size for static pages suggests that for large documents transmission is begun before the full document is read from disk. The linear dependence of request processing time with document size for dynamic pages suggests that dynamic pages are fully constructed before transmission is begun. The average request processing time of 400-500KB dynamic pages is 3 times as much as that of 0-100KB dynamic pages, and 10 times as much as that of 400-500KB static pages. This further explains the difference between average request processing time for static and dynamic pages as shown in Figure 5.

4.6 Response Transmission Time and Throughput

Figure 10 shows that the response transmission time grows proportionally to the response transmission size. This agrees with our expectation. However, this figure does not provide the details for the transmissions whose sizes are below 100KB. These transmissions are actually the most common cases as shown in Table 11. Should the conclusion be different for them?

To answer this question, we measure the response transmission throughput over a wide range of response transmission sizes. Figure 11 shows that the smallest transmisions

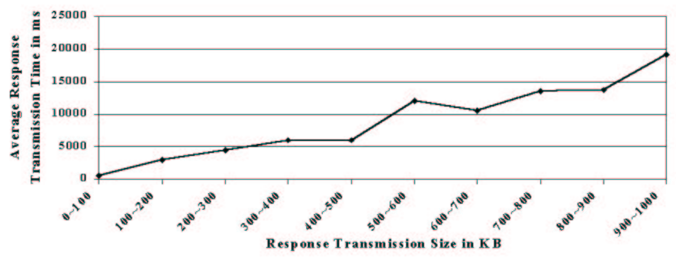

Figure 10: Response Transmission Time vs. Response Transmission Size

Response Transmission Size	%
Below 2KB	9.9%
2-5KB	14.6%
5-10KB	19.3%
10-20KB	23.9%
20-50KB	24.9%
50-100KB	5.6%
Above 100KB	1.8%

Table 11: Occurences of Response Transmission Sizes in Percentage

have the best performance, and the common medium-sized transmissions (5-20KB) have the worst performance. We speculate that this phenomenon is related to the transmission mechanisms of TCP. Usually, small pages that involve only a couple of roundtrips are dominated by the TCP slow start phase. They do not have enough time to detect and fully utilize the network capacity. Bigger files, on the other hand, can get over the slow start, and enter the period in which the TCP congestion control stablizes the transmission. This explains the rising half of the curve. To explain the falling half of the curve, we need to consider two cases. When a number of packets are sent back to back in a single roundtrip, there will be an instant peak throughput. For tiny pages, this is the final throughput, since they can be contained in the very few packets that are sent together even in the slow start phase. When packets are sent in separate and small groups with long intervals in between, the throughput should be very low. This is the case for medium pages.

To verify our speculation, we try to charaterize the response transmission pattern and see if pages are mostly sent with a “slow” start, a “fast” start, or a “random” start. To do this, we define the estimated roundtrip time to be either half of the connection time or the typical 100ms cross-country roundtrip time, whichever is smaller. If we see a packet interval bigger than the estimated roundtrip time, we determine that the pair of packets are sent in two different roundtrips. This allows us to distinguish the packets in the initial burst. Figure 12 shows a uniformly good adoption of ”slow” start across all types of pages. From

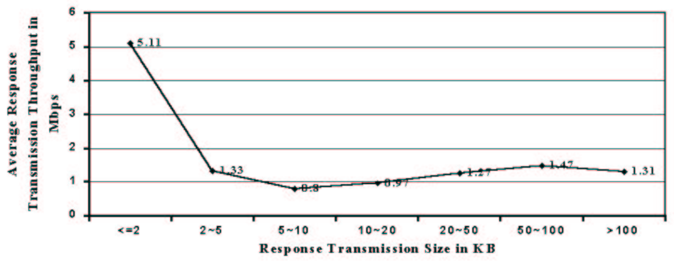

Figure 11: Response Transmission Throughput vs. Response Transmission Size

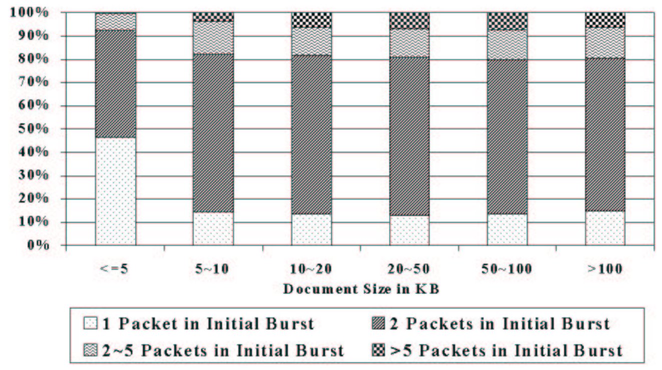

Figure 12: Number of Packets in Initial Burst

this figure, we also find a response transmission problem in the current practice. About 13% of the pages have only one packet in their first roundtrip. If the client side enables delayed-ACK, this single initial packet will not be acknowledged until the client has waited the full 200ms delay for a second packet, thus causing a 200ms delay in the next roundtrip.

5 Protocol Compliance

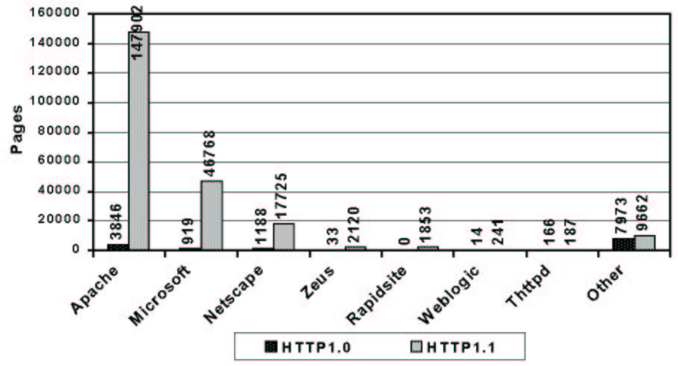

Figure 13: Use of HTTP/1.1 and HTTP/1.0 in Major Web Servers

Figure 13 shows the current use of HTTP/1.1 and HTTP/1.0 in major Web servers. Based on the protocol records of all the successful Web transactions, we have the following results.

(1) The syntactic requirements are fulfilled by most servers, but there do exist a number of serious incompliances
(2) Persistent connections are widely supported, but its announcement is sometimes confusing
(3) Conditional GET, the fundamental technique for Web caching, is not sufficiently supported
(4) Some communication features are frequently exploited, while others are rarely touched

In the rest of this section, we will present our analysis only of static pages, because cases are similar for dynamic pages.

5.1 Syntactic Requirements

Web Server	Apache	Microsoft	Netscape
Count	48	2014	11

Invalid End-of-Line Marker (Total = 2811)

Web Server	Apache	Microsoft	Netscape
Count	712	1436	315

Non-absolute URL (Total = 2729)

Table 12: Non-compliance to Syntactic Requirements

Table 12 shows two common incompliances to syntactic requirements. In both HTTP protocols, the end-of-line marker CRLF is the delimiter of the message header and the message body. It is also the delimiter of the fields within the message header and the delimiter of the multiparts within the message body. However, some servers use invalid end-of-line markers such as bare CR and bare LF. This makes it hard for the client to distinguish the different structures in the response message. Another common syntactic requirement in both HTTP protocols is the use of an absolute URL in the "location" header field. However, some servers use a URL that is relative to a local one. This forces URL bookkeeping by the client to translate the relative URL into an absolute one. We also encountered 62 pages that had no protocol version number in the message header. Due to this, we were unable to predict the protocol behavior of those servers and take the appropriate actions. The last problem we find is that some servers do not recognize the requested URL in the absolute URL format, although this is required for transition to future protocol versions. We re-crawled 5000 static pages twice, using relative requested URLs the first time and absolute requested URLs the second time. It turned out that 504 pages accepted the first but refused the second.

5.2 Performance Features

5.2.1 Persistent Connections

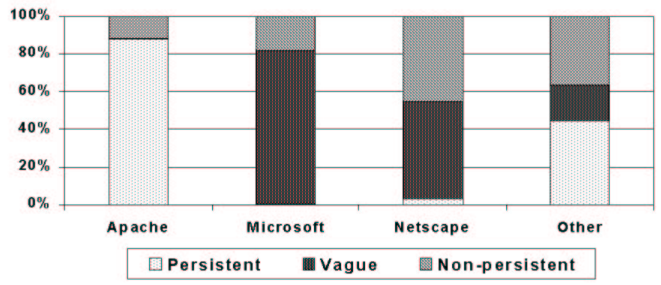

Figure 14: Announced Support for Persistent Connections by HTTP/1.1 Servers

Figure 14 shows the announced support for persistent connections by HTTP/1.1 servers. Apparently, this displays heterogeneous behaviors across different servers. Most of Apache servers claim persistent connections, some of them claim non-persistent connections, and few of them are vague by not indicating anything. Few of Microsoft-IIS servers and Netscape-Enterprise servers claim persistent connections, and most of them are vague. Our study of this phenomenon focuses on the vague class.

Web Server	Apache	Microsoft	Netscape
Vague	330	38082	9190
Vague and Alive	287	37970	9073

Table 13: Vague and Alive Connections by HTTP/1.1 Servers

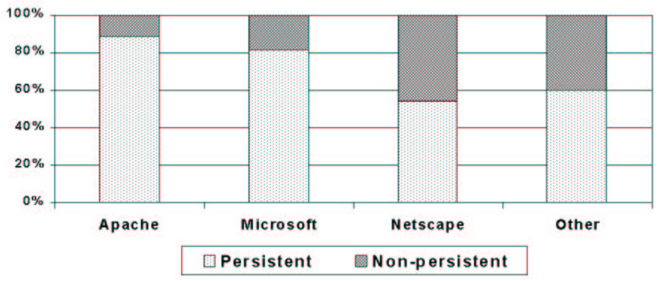

Figure 15: Actual Support for Persistent Connections by HTTP/1.1 Servers

Table 13 provides the number of pages for which the connections are still alive after the transmission has been completed, out of the number of all vague pages. This shows that, in most cases, HTTP/1.1 servers mean "persistent" by being vague. Therefore, accounting for this, we re-draw Figure 14 as Figure 15. This time, the figure displays homogeneous behaviors across different servers. The difference between the two figures shows that Apache servers indicate persistent connections explicitly, while Microsoft and Netscape servers indicate persistant connections by not denying persistance. But these different behaviors when encountered by the client would be very confusing.

Another interesting observation is that a lot of HTTP/1.0 servers also claim persistent connections despite that this feature is formally specified only in HTTP/1.1.

5.2.2 Conditional GET

Page Status	Count	
Modified	352	
Not Cachable	204	
Unmodified and Cachable	Message Header Returned	2532
	Full Document Returned	1911

Table 14: Responses for Conditional GET Requests

Our actual crawl always use the unconditional GET method. However, a problem shows up when we try to update our data using the conditional GET method. Many servers do not recognize the time validator in the conditional GET message, and return the full document even if it has not changed. Therefore, we re-crawled 5000 static pages twice, using the unconditional GET the first time, and the conditional GET immediately after. Table 14 shows that among the 4443 unmodified and cachable pages for which only message headers should be returned, 1911 pages were actually returned in full documents.

5.3 Communication Features

Feature	%	Feature	%
Content Type	99.9%	Content Location	6%
Date	99.5%	Expires	5%
Server	98.5%	Vary	1%
Content Length	62%	Content Language	0.3%
Last Modified	58%	Content Encoding	0.1%
ETag	52%	Allow	0.1%
Cache Control	11%	Content MD5	0.0%

Table 15: Usage Percentages of Communication Features

Feature	% in HTTP/1.1	% in HTTP/1.0
Keep Alive	3%	60%
Pragma	6%	1%
Content Base	0.0%	0.0%

Table 16: Usage Percentages of Legacy Features

Table 15 shows the sorted usage percentages of communication features. Some features are frequently exploited for better communication, while others are rarely touched.

6 System Suggestions

Based on our characterizations of the Web, we have the following suggestions for Internet improvements.

6.1 DNS Negative Caching Revalidation

Danzig [5] showed that DNS negative caching could achieve only marginal performance improvement on the root DNS servers. However, we would like to advocate this mechanism for Web transactions with two arguments. First, as is shown in Section 3, DNS failures account for 50% of the communication failures. We believe that this number will steadily increase in the future. This is because with the tremendous growth of the Web, the maintenance work on the Web will become more and more challenging, resulting in more and more invalid host names. Second, as is shown in Section 4 and Section 3, the average DNS time of successful Web transactions is 359ms for static pages and 233ms for dynamic pages, while the average time for a DNS failure to return is 2 seconds. Therefore, if the negative results are cached in the local name server, not only the amount of DNS traffic will be greatly reduced, but also on average the perceived latency of the return of DNS failures will be much shorter.

6.2 URL Negative Caching on Extensible Routers

As is shown in Section 3, "Not Found"s account for 90% of the transaction failures. We believe that this number will also increase in the future due to the increased difficulty in maintaining the Web. In the light of DNS negative caching, we propose the investigation of URL negative caching on extensible routers to resolve this problem. Specifically, extensible routers, which understand the HTTP protocols, can record the unsuccessful URLs to prevent future requests. We believe that this mechanism will save a lot of network traffic and provide much faster response to invalid requests.

6.3 Tradeoff of Redirections

In the original design of the HTTP protocol, a redirection message is used to convey that the document has been moved. But today, people use redirection messages to implement load balancing by switching the jobs on a busy server to an idle one, to delay the request processing by redirecting to the same place, and to gain more hits on the site by wiring to other sites through local URLs. As is shown in Section 4, these strategies deteriorate the performance to a large extent. Therefore, if performance is an important metric in the site design, conservative use of redirections should be considered.

6.4 Possibility of Reducing Request Processing Time

As is shown in Section 4, the linear dependence of request processing time with document size put large dynamic documents at great disadvantage. Therefore, if parallelization of dynamic page construction and transmission is possible, significant enhancement will be achieved.

6.5 Discreet use of TCP Slow Start on 5-20K Pages

As is shown in Section 4, the common 5-20K pages have the lowest response transmission throughput due to the effect of TCP slow start. Therefore, we suggest that servers differentiate medium pages from small and large ones when using the TCP slow start. A "faster" start for medium pages may achieve much better performance.

6.6 Improved Use of Protocol Performance Features

As is shown in Section 5, different servers claim persistent connections in different ways. This may cause confusion in protocol communication, compromise the potential benefit of this performance feature, and cost more implementation efforts to accomodate the variances. In the same section, we also point out that the support for conditional GET is not strong. This will directly limit the applicability of Web caching. Based on these two problems, we call for better use of protocol performance features.

7 Related Work

We compare our work and previous efforts in two primary aspects, the methodology of measurement and the object of investigation.

The methodologies for Web measurements can be classified into two categories. The first one conducts a passive analysis on traces, which can be from Web servers [1] or local area networks [6]. A deficiency with the passive measurements is the lack of flexibility as they completely rely on the existing information resources. The second category, which includes VerMIC, conducts an active probe by injecting stimulus into the Web. For example, Keynote Systems [11] and Krishnamurthy [9] set up a number of testing agents in a wide area and issue HTTP requests to the designated Web sites. This fixed set of Web sites limits the scalability of these studies in comparison to our use of VerMIC. A seeming problem with VerMIC is its location bias resulting from its single point of initiating requests rather than having testing agents in several locations. However, we argue that, assuming the Web is well distributed, its general characteristics observed at all locations should be homogeneous.

The Web has been the object of many studies, but only some aspects of our investigation have been addressed previously. There have been no previous formal studies of failures in the communication phases of a Web transaction. The interests of past work in performance have been focussed on the response transmission phase, specifically bulk throughput, packet delay, packet loss rate, and bandwidth between hosts. No earlier study identifies the empirical parameters of all the individual communication phases and synthesizes them to produce an intergrated view of the partition of their impacts, as is done here. Krishnamurthy [9] presented a systematic analysis of protocol compliance, and directly motivated the protocol compliance part of our work. We have made new discoveries in this area, taking advantage of our extensive and scalable infrastructure.

8 Conclusion

In this paper, we characterize the Web in three aspects: Web failures, timing performance, and protocol compliance. Our results capture the current status of the Web, and have implications for system improvements. However, our study still leaves open a number of interesting questions. The potential of DNS negative caching and URL negative caching still requires more investigation. The feasibility of parallelization of dynamic page construction and transmission for heterogeneous request processing programs is still not clear. And some of our observations can't be explained. We hope future work will give answers to them.

References

[1] Martin F. Arlitt and Carey L. Williamson. Web server workload characterization: The search for invariants. In *Proceeding of the 1996 Conference on Measurement and Modeling of Computer Systems*, pages 126–137, 1996.

[2] Hari Balakrishnan, Mark Stemm, Srinivasan Seshan, and Randy H. Katz. Analyzing stability in wide-area network performance. In *Proceeding of the 1997 Conference on Measurement and Modeling of Computer Systems*, pages 2–12, 1997.

[3] Paul Barford and Mark Crovella. Measuring web performance in the wide area. Technical Report BU-CS-99-004, Boston University, Computer Science Department, 1999.

[4] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[5] Peter B. Danzig, Katia Obraczka, and Anant Kumar. An analysis of wide-area name server traffic. In *Proceeding of the 1992 Conference on Communications, Architectures and Protocols*, pages 281–292, 1992.

[6] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey Mogul. Rate of change and other metrics: a live study of the world wide web. In *Proceeding of the 1997 USENIX Symposium on Internet Technologies and Systems*, 1997.

[7] J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1*, 1999.

[8] Van Jacobson. Congestion avoidance and control. In *Proceeding of the 1988 Symposium on Communications, Architectures and Protocols*, pages 314–329, 1988.

[9] Balachander Krishnamurthy and Martin Arlitt. PRO-COW: Protocol compliance on the web—a longitudinal study. In *Proceeding of the 2001 USENIX Symposium on Internet Technologies and Systems*, 2001.

[10] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400:107–109, 1999.

[11] Chris Loosley, Richard L. Gimarc, and Amy C. Spellmann. E-commerce response time: A reference model. In *Proceeding of The Computer Measurement Group's 2000 International Conference*, 2000.

[12] Stephen Manley and Margo Seltzer. Web facts and fantasy. In *Proceeding of the 1997 USENIX Symposium on Internet Technologies and Systems*, 1997.

[13] Henrik Frystyk Nielsen, James Gettys, Anselm Baird-Smith, Eric Prud'hommeaux, Håkon Wium Lie, and Chris Lilley. Network performance effects of http/1.1, css1, and png. In *Proceeding of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1997.

[14] Vern Paxson. End-to-end routing behavior in the internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, 1997.

[15] Vern Paxson. End-to-end internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, 1999.