The Effect of Early Packet Loss on Web Page Download Times

James Hall, Ian Pratt, Ian Leslie and Andrew Moore
University of Cambridge Computer Laboratory
JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom
{firstname.lastname}@cl.cam.ac.uk

***Summary*—**

Identification of the various elements contributing to the download time of Web objects has been the subject of much research. It is, however, the download time of a set of objects comprising an entire page which is of critical importance to the user — particularly in the presence of delay leading to subjectively long waits. Because browsers typically use parallel TCP connections to download the set of objects neither the overall page download time, or the contribution of delay, are simply the sum of the individual object download times or delays. We present a technique for identifying and examining the complete set of connections involved in a page download which then enables us to fully analyse the contributory time components and to assess the potentially serious delays which arise from loss early in the connections' lifetimes.

I. Introduction

No one who has used a browser to access pages from the World Wide Web has escaped the experience of overlong download times. While times in the order of seconds are acceptable, those exceeding even a few tens of seconds become increasingly annoying.

Page downloads using HTTP/1.0 [1] suffer from the problem that each constituent object must be downloaded using a new TCP connection. Each object download is thus subject to TCP connection latency and commences under slow start conditions. Additional load is placed upon the network, and the server has to service and maintain a large number of individual connections.

Improvements to overcome the shortcomings in HTTP/1.0 were suggested by Mogul and Padmanabhan [2]. Of particular relevance were *persistent* HTTP connections (P-HTTP) and *pipelining* of multiple object requests over a single connection. Experimental data supported their recommendations and showed a significant reduction in overall latency for retrievals. A following paper by Mogul [3] demonstrated the significant reduction in latency that would be achieved by using P-HTTP and pipelining; extensive simulation also demonstrated the reduction in use of server resources under TCP TIME-WAIT states of various durations.

HTTP/1.1 [4] published in January 1997 incorporated persistent HTTP connections and pipelining, together with a negotiation protocol allowing for persistence negotiations between HTTP/1.0 and HTTP/1.1 clients and servers.

Nielson et al. [5] conducted a series of experiments using links of different bandwidths between a server/client pair set up for the purpose of evaluating the performance of P-HTTP and pipelining and concluded that significant improvements in HTTP performance required the use of both techniques. It was noted, however, that such improvements were of less significance when data was carried over a slow speed link (i.e., PPP over a dial-up link). This observation was reinforced by Heidemann [6] who showed, as a result of analysis of traffic models, that lowest link bandwidths in excess of 200 Kbps are required in order to realize a subjective improvement in performance seen by the user. He reported validatation of the models used in [7] by comparison with traces of real traffic. The paper also identifies some areas of TCP/HTTP interaction where persistence introduces delay.

Identification of the various elements contributing to the download time of Web objects has been the subject of much research. It is, however, the download time of a set of objects comprising an entire *page* which is of critical importance to the user — particularly in the presence of delay leading to subjectively long waits. Because browsers typically use parallel TCP connections to download the set of objects, and because they may not use these connections in a timely manner, overall page download times are not simply the sum of the individual object download times, and delays at the page scale are similarly not the sum of delays suffered by individual objects. To fully analyse the time components of page downloads it is therefore necessary to consider the full set of constituent object deliveries, the connections used, and their interrelationships over the total download period. We must distinguish the download of a Web *page* from that of its component *objects*.

We illustrate this point with Figures 1 – 3 which show three simple cases of page downloads and the effects that delayed objects might have on overall times. In these fig-

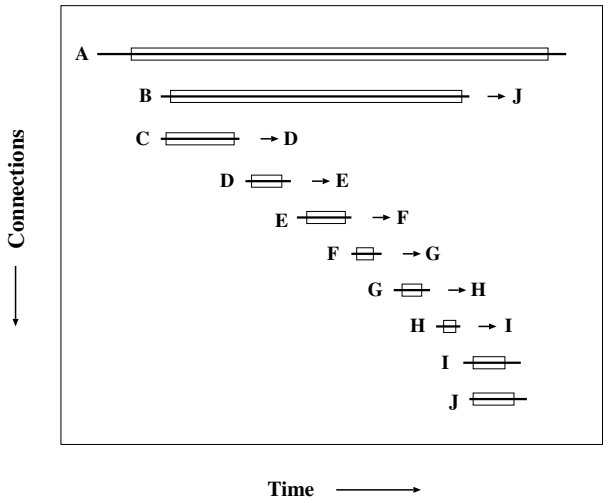

Fig. 1. Page download with object B delayed — the whole page download is not delayed

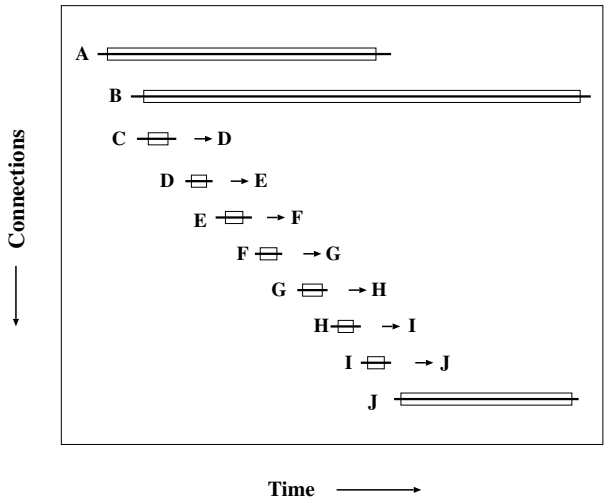

Fig. 2. Page download with objects B and J delayed — the whole page download is delayed

ures each single horizontal line represents the period of a TCP connection from the client's SYN, and the superimposed boxes the period between HTTP request and the last data packet of the response. In each figure Object A is an HTML document and Objects B – J might be small in-line image objects. The X-axis represents elapsed time, and the Y-axis the order in which connections are opened and the objects requested. Typical browser behaviour is represented in which the root document is downloaded on one connection, and a maximum of two concurrently open connections is maintained for the download of subsidiary objects. The arrowed letter following each connection line indicates the connection which follows its closure.

In Figure 1 Object B is delayed, but with no effect on overall page download time. Objects D – J are delayed as only a single connection is available, but not by the magnitude of the Object B delay; this will be noticeable in a browser which renders objects as they are received.

In Figure 2 Object B is delayed beyond the download time of the container document, and hence adds to the overall page download time. Objects C – I are delayed as in Figure 1. Object J is also delayed, but adds nothing to the delay in overall time already contributed by Object B. Although the overall page download time has been extended, it is only the rendering of two objects which is significantly delayed.

Figure 3 also shows a download in which two objects are delayed. In this case the downloads of six of the remaining objects are also significantly delayed resulting in late receipt of eight of the nine images contained in the page.

It will be appreciated that the typical Web page will probably contain many more objects than in the simple examples shown, and that, while individual object delays

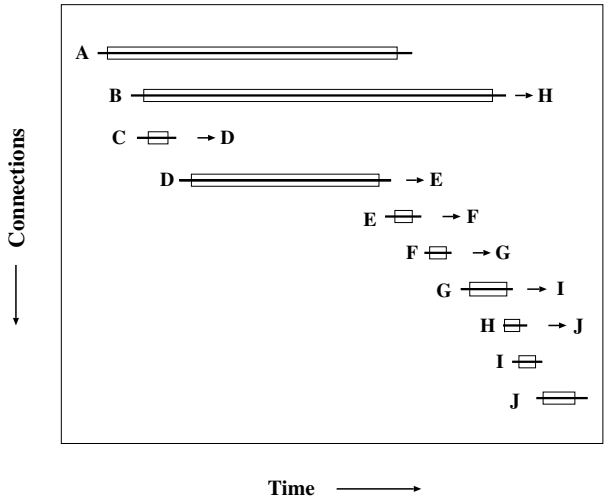

Fig. 3. Page download with objects B and D delayed — the whole page download is delayed

of the general pattern shown in Figures 1 and 2 may serve only to introduce minor delays in the presentation of a whole page, there is also the possibility that delays occurring in patterns similar to that shown in Figure 3 may result in very a significant cumulative degradation in performance.

In the initial stages of a TCP connection neither client nor server has seen enough packets to establish a usable round trip time estimation. In the case of packet loss retransmission will therefore take place only after a default timeout period (typically in the order of 3 seconds increasing exponentially for each repeat). We refer to packet losses during this period as *early* packet loss. Such delays in individual connections have been noted by the WAWM Project [8].

We note in this context a serious implication of the use

of many short-lived TCP connections delivering single objects. While long delays may be introduced by early packet loss on any connection, the page downloads using non-persistent connections are particularly vulnerable — most or all activity taking place in conditions of potential early packet loss.

While the delay contribution to page download times due to connection setup latency, or to packet loss on an established connection, are likely to comprise a small number of round trip times (RTTs) — in the order of some tens or hundreds of milliseconds — those due to early packet loss contribute delays in the order of seconds, and which, when occurring in patterns similar to that illustrated in Figure 3 may reach the order of tens or even hundreds of seconds.

In the rest of this paper when we refer to 'delays', either in reference to pages or single TCP connections, we mean delays arising from early packet loss. It is the effect of these *gross* delays on whole page download times which we investigate in this study.

In the next section we consider the effects of early packet loss on page downloads. Section III explains how we calculate the degree of delay and characterize the distribution of the contributory individual object delays. In section IV of this paper we consider some of the difficulties encountered in collecting the data needed to examine whole page retrievals, and describe our method of data harvest. We give a detailed analysis of downloads from a site known to suffer serious delay in section V and go on in section VI to place this in the context of a wider sample of Web traffic. Section VII considers the implications of early packet loss for persistent connections.

II. The effects of early packet loss on TCP/HTTP connections

We stated in the introduction that we are particularly concerned in this paper with examination of the effect on page download times of *gross* delays (i.e., in the order of seconds upwards) caused by loss of packets before the TCP connections carrying HTTP requests and responses have established an RTT for use in retransmission timeouts. It is also inherent in this stage of a connection's lifetime that fast-retransmit mechanisms will not be operative. In the absence of a reliable RTT value the majority of TCP implementations will employ a default timeout in the order of three seconds. It is also worth noting that in the context of many short-lived connections default retransmission values based upon shared routing metrics are also unlikely to be available to individual connections.

On an individual connection such packet losses will manifest themselves as:

- Loss of client's SYN packets(s) causing one or more SYN retransmissions.
- Loss of server's SYN ACK causing one or more SYN retransmissions.
- Loss of client's request causing one or more retransmissions.
- Loss of server's ACK to the request causing one or more retransmissions.
- Loss of server's response.

Such losses may occur on both persistent and non-persistent connections, but in the persistent case losses of requests/responses after the first will normally cause retransmits based upon established RTT values or fast-retransmit mechanisms. The case where a client's first request is lost can be distinguished from that where the server merely exhibits a long delay before responding by the receipt of an ACK for the segment carrying the request.

In order to study this phenomenon we analyzed three traces, each of approximately two hours duration, of all traffic to and from the University site, identifying the TCP/HTTP connections exhibiting this type of delay. Reconstruction of the entire page downloads of which such connections formed a part then allowed us to calculate the overall effect of all delays on the total page download time.

In Tables V and VI we summarize our observations for each of the traces: Table V records the numbers of clients, servers, connections, downloaded objects and pages observed in each trace, together with the proportion of connections upon which multiple requests were issued. In Table VI, we show, in columns 2 – 4 the proportion of connections and pages subject to the delays in which we are interested, and of the servers involved. Columns 5–9 show the incidence of connections upon which we observe delays as represented by seeing retransmissions of client SYN packets or first requests, or by failure to see any request, response, or to establish a connection. In all of the last three cases the browser counts a connection as open and the opening of further connections is inhibited. Finally in columns 10 and 11 we show, for comparison purposes, the proportions of later packets (i.e., on an established connection) which are retransmitted.

III. Calculation of whole page delay

A. The degree of delay

We define whole page delay as the difference between the time when delivery of the last object of a page completes and the corresponding time had there been no delays in the delivery of individual objects. Calculation

Fig.	Delayed Object				Total	Page Download Time		*cod*
	Object	Delay	Object	Delay	delay	Delayed	Undelayed	
1	B	50	-	-	100	100	100	10
2	B	130	J	90	310	180	100	31
3	B	130	D	60	670	150	100	67

TABLE I

SUMMARY OF DELAYS AND CUMULATIVE OBJECT DELAYS FOR THE PAGE DOWNLOADS ILLUSTRATED IN FIGS. 1 – 3

of this delay therefore depends upon establishing the notional download time without delay.

Even with the relatively rich set of data at our disposal it is not entirely simple to derive a model of an equivalent, non-delayed, set of downloads for a page. It can be difficult to ascertain the target number of concurrently open TCP connections: browsers are often inconsistent — even within a single page, the point of connection closure recognized by browsers may also vary (the completion of the full TCP 4-way close, the transmission of the browser's FIN packet, the receipt of the server's FIN packet on a non-persistent connection, etc.), the opening of new connections may be delayed by the browser, and account must be taken of the delay between causative and consequent events as seen by the probe.

For each object we know the timing of events at both the TCP and HTTP/HTML levels of the protocol stack. Where delays occur we can therefore calculate the delivery period of the object and the duration of the connection with the removal of the delay.

A frequency analysis of the number of connections currently open, as each fresh connection opens, is calculated on the basis of the various possible criteria which a browser may use. This gives us an indication of the criterion used by the browser and its concurrency target.

We reconstruct the whole page download by taking the set of connections used, discarding any which did not establish a full connection or succeed in obtaining a server response, and correcting the remainder for any delay. The modified open and 'close' times can then be calculated for each successive connection on the basis of the timings of the preceding connections and the browser's concurrency criterion and target. We assume that any inconsistencies in the browser's observed pattern of connection concurrency would also apply to the recalculated set of download connections [1] , and in calculating when each connection would open we therefore allow the number of currently open connections at the point of opening to maintain inconsistencies. We also make the assumption that any delays in the server remain constant despite changes in the pattern of requests over time, and similarly that any browser-introduced delays remain constant. Because we know the relationship between the links contained in the root document and object downloads we are able to ensure that modified connection timings do not cause an object to be requested before the browser would have seen the corresponding link.

In previous work [9] we describe the techniques that we use to estimate server and client delays together with the (partial) RTTs between probe/host/probe which establish the time lag between causative and consequent events, but for the purposes of this calculation such techniques are not considered necessary if we assume that client/server delays and network transit times remain unchanged. We consider that the assumptions we make are reasonable, as our new model of the set of downloads without delays does not in general produce major changes to either patterns of network or server activity. Any variations which break these assumptions are also likely to be at least two orders of magnitude smaller than the phenomena that we are studying.

B. Distribution of delay

From the subjective perspective of the user it is not just the overall page download time which is important, but also the distribution of any object delays within the download period. The delays shown in Figure 2 would probably be perceived as more annoying than those shown in Figure 3, although both pages may be completed in the same times. We assume that in the general case in-line objects are fetched in the order that the browser's HTML parser encounters the relevant links, and that this order reflects the significance and rendering order of the objects.

In our analysis we have characterized the distribution of individual object delays using a simple metric — a *cumulative object delay* (*cod*) for a page of N objects calculated as follows:

[1] There appears to be no correlation between unsuccessful or delayed connections and such inconsistencies

$$cod = \left(\sum_{n=1}^{n=N} (D_n - U_n) \right) \times \frac{1}{N}$$

where:

N is the number of objects in the page,
D_n is the delivery completion time of object n and
U_n is the notional delivery completion time of object n in the absence of delay

As an illustration of the effect of object delays and distribution on the *cod*, Table I summarizes the downloads shown in Figures 1 – 3, giving the *cod* for each case, assuming the container document downloads in 100 time units and that each image object would download in 10 time units if not delayed. We stress that the *cod* represents a measure of the distribution of delay; while it is intuitive that delay early in the download of a page is subjectively less acceptable than delay affecting only the later components, assessment of user dissatisfaction is not within our field of study.

IV. Data collection for the study of whole page downloads

The collection of the data required to study whole page downloads presents a challenge. While it is comparatively easy to gather data about individual TCP connections, or Web pages/objects it is very much more difficult to amass this data in a way which also allows its integration into a whole. The objects comprising each page must be identified in order that the network activity involved in their individual downloads, and the relationships between the connections used, can be established.

Server logs will normally yield some of the required data — in terms of the objects downloaded, when and to which clients — but do not normally record the level of detail of network activity required or the interrelationships between the objects which they host. It would be possible to scan the contents of hosted objects and establish relationships, but such scans and the logging of additional network detail at the packet level would entail considerable extension of the normal logging mechanism.

Browsers could be instrumented in order to collect the required data, but this would be a non-trivial exercise, and it would be unlikely that enough modified browsers could be deployed to provide either a large or representative sample of Web activity. While servers with enhanced logging facilities would provide a very large sample in volume terms it is again unlikely that deployment would be sufficiently wide to provide a representative sample.

All of the relevant data can be collected from the network itself, hence allowing larger and more representative samples, using passive monitoring techniques. There is a considerable and diverse body of research in the area of network monitoring and the analysis of collected data in the context of TCP/HTTP and Web Server performance. We mention only that which is most relevant to our current work.

The ubiquitous Tcpdump and its variations remain the staple collection tool for much post-collection analysis based research; despite its high copy overheads and inability to keep pace with high arrival rates, its packet filtering capacity and familiarity are attractive to many projects. More recent monitoring designs, aimed towards keeping pace with today's network bandwidths, include the ambitious OCxMon architecture [10] based upon 'off the shelf' components, and the AT&T Labs PacketScope [11] probe based upon a dedicated 500-MHz Alpha workstation.

The interaction between TCP and HTTP, and the TCP behaviour of Web Servers, is studied using active techniques with TBIT [12] and in the WAWM project [8] using a combination of passive/active monitoring techniques.

The collection and integration of data derived from multiple levels in the network protocol stack is the subject of a growing research field, including the Windmill monitoring architecture [13] with which integrated data was used to identify causes of routing instability [14], and in the BLT project [15] where TCP/HTTP phenomena are studied using data extracted from the headers of both protocols contained in packets collected using the Packetscope.

We have collected the data upon which this study is based using our own Nprobe passive network monitoring probe [9][16]. Nprobe collects data from several levels of the stack and is stateful (i.e., is able to associate the individual packets that it sees passing on the network into flows defined at the network and transport levels of the protocol stack, and into Web transactions at the HTTP level). It is therefore capable of integrating the required data while achieving a high level of data reduction.

For the purpose of this exercise we used data extraction modules recording data contained in the relevant fields from packet headers and the contained HTTP request and response headers. We also deployed a module which extracts the links contained in HTML documents. During the post-collection analysis of the collected data we were therefore able to examine activity at the network level and to precisely associate it by objects fetched, and by page.

While assumptions can be made about the relationship between the objects seen using a less comprehen-

sive set of data (e.g., it could be inferred by recording HTTP response content fields that for each pair of hosts all GIF or JPEG objects were associated with the immediately preceding HTML document) there is considerable scope for error. The contents of concurrently downloading pages can be confused, re-fetched components of revisited pages are not distinguished, subsidiary container documents (e.g., frames) are not identified, and Web pages increasingly span content originating from multiple servers. By using referrer data together with knowledge of which objects are linked, the type of link, and when the browser 'sees' each link, we are able to form a very much more accurate model of the relationships between objects.

Our probe was fed from the switch connecting the University network to the United Kingdom SuperJANet network, and hence saw all traffic between the site and the rest of the world. Traces 1 – 3 examined in section VI are each of approximately two hours duration, collected at around mid-day on a week day in each case. Trace 4, analyzed in section V, is of traffic seen between approximately 11.30am and 1.50 pm extracted from a longer trace of 24 hours duration. In Tables V and II we summarize the Web traffic seen in each of the traces.

V. Analysis of page delays seen in traffic to a single site

Users of a popular British news Web site frequently experience long and frustrating delays in downloading pages. We consequently monitored HTTP traffic between this site and our university network for a period of 24 hours. Analysis of the trace revealed that downloads from the site suffered a high incidence of exactly the delays forming the subject of this paper, and that the delivery of many pages was thereby considerably delayed.

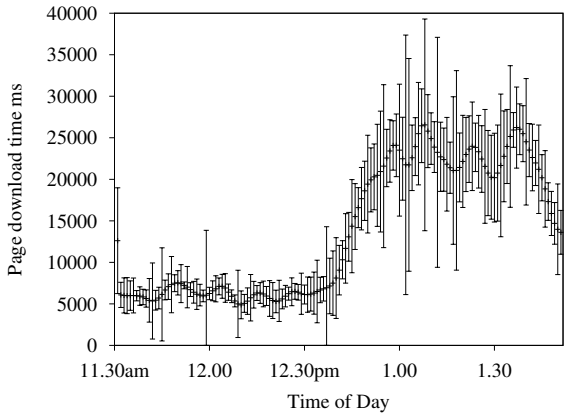

Fig. 4. Whole page download times from the news server

We present here our analysis of the trace period between approximately 11.30am and 1.50pm. This period is of particular interest as it spans the beginning of the lunch time period when the server load may be expected to increase as people visit the site during their lunch break. Our results are summarized in Tables II and III. The entry for Trace 4 shows details of the whole trace, traces 4a and 4b show the trace partitioned into the periods 11.30am – 12.40pm and 12.40pm – 1.50pm.

In Figure 4 we show the download times of the pages seen averaged over periods of 60 seconds. The error bars show the standard error of the mean for each period. It can clearly be seen that page download times increase dramatically between approximately 12.30pm and 1.00pm as load increases. At the higher page download times the error bars show a corresponding increase in the standard error: the difference in download times between delayed and non-delayed pages becomes more marked due to the magnitude of the delays.

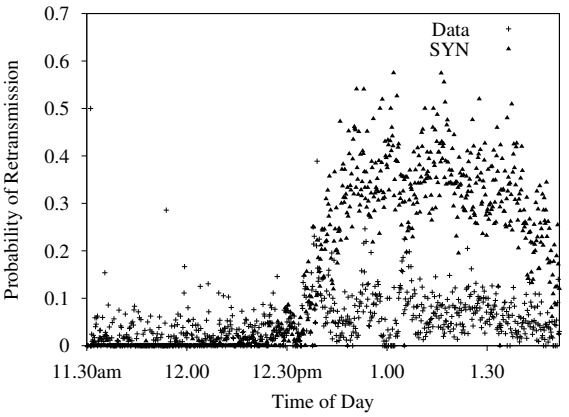

Fig. 5. Averaged probability of SYN and data packet retransmission on connections to the news server

Figure 5 presents the probability, averaged over 10 second intervals, of either SYN or data packet retransmission on individual TCP connections. The probability that data packets will be retransmitted due to network loss rises slightly from approximately 12.00pm onwards as (by inference) traffic levels increase. We suggest that the contrasting very sharp rise in the probability of SYN retransmission at approximately 12.40pm is due to connection refusal by the server as load increases. We show in [9] that from 12.40pm onwards server latency remains approximately constant, although higher and more dispersed than during the earlier period of lower demand, and that this time therefore represents the point at which the server becomes saturated and is, in effect, exercising admission control through connection limiting. It is the sharp rise in 'lost' SYNs that is the principal cause of the corresponding jump in page download times.

Figsures 6 and 7 show the distribution of whole page download times for the periods 11.30am – 12.40pm and

Trace No.	Number of Clients	Servers	Conns.	Objects	URLs	Pages	% Delayed Conns.	Pages	Servers	% Persistent Connections
4	732	29	89195	92901	1570	10232	14.39	18.82	34.48	0.52
4a	407	18	43440	44643	835	4908	2.64	5.03	33.33	0.70
4b	470	24	45818	48204	1098	5327	25.52	31.31	41.67	0.36

TABLE II

SUMMARY OF TRACE 4 — TRAFFIC TO AND FROM THE NEWS SITE. TRACES 4A AND 4B ARE THE FIRST AND SECOND PERIODS OF TRACE 4

Trace No.	% of connections subject to early loss					% Early packet loss		% Later packet loss	
	Client		Other						
	SYN	Req.	No Req.	No Rep.	No conn.	SYN	Req.	Client	Server
4	13.18	0.86	1.39	1.62	1.07	119.42	16.67	0.12	2.32
4a	2.05	0.35	0.26	0.39	0.14	102.50	4.77	0.09	2.25
4b	23.73	1.35	2.46	2.79	1.94	135.22	27.78	0.15	2.38

TABLE III

SUMMARY OF DELAYS: TRACE 4 — TRAFFIC TO AND FROM THE NEWS SITE

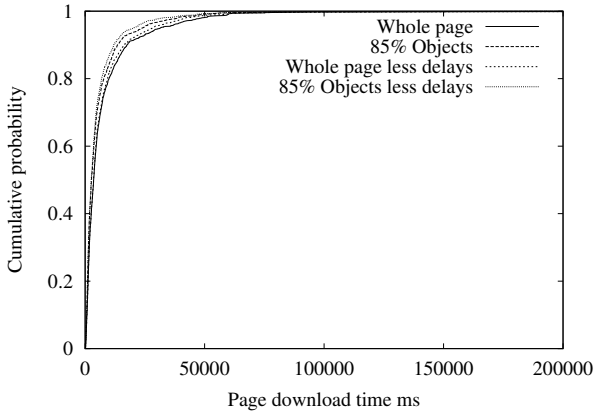

Fig. 6. CDF of page download times from the news site: 11.30am – 12.40pm

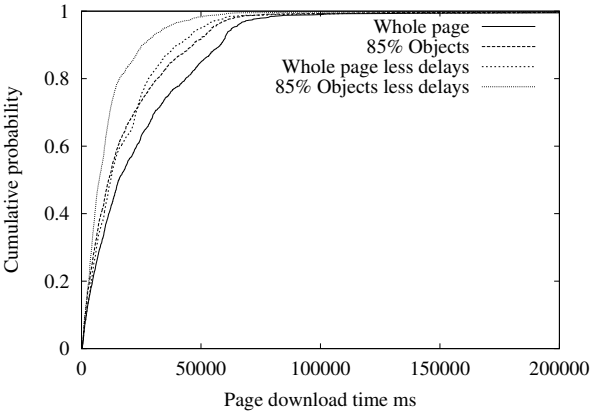

Fig. 7. CDF of page download times from the news site: 12.40pm – 1.50pm

12.40pm – 1.50pm respectively. In each case we show the distribution of times for the whole page and for 85% of its content to be downloaded. We also show the corresponding distributions with the effect of delays removed. The difference between the two periods is marked; the download times for the 75th and 90th percentiles of all pages is summarized in Table IV

Figure 8 shows the CDF of the degree of delay within each of the various manifestations of early packet loss seen in our trace. Note that, with the exception of delays caused by single SYNs not receiving a response, or connections upon which the client issued no request — in neither case is packet retransmission involved — the reliance of retransmission on default values is very clearly seen from the distinct steps at 3, 6, 9... seconds.

VI. ANALYSIS OF PAGE DELAYS OVER ALL TRAFFIC MONITORED

The high rate of SYN loss observed in the traces of traffic to the news site described in Section V is fortunately not typical of the majority of page downloads. In Tables V and VI we present summaries of three traces, each of approximately two hours duration, collected at the

Trace No.	Number of Clients	Servers	Conns.	Objects	URLs	Pages	% Persistent Connections
1	19077	10820	827720	1608944	157146	368774	15.49
2	18510	13592	881719	1788477	212412	572355	16.03
3	10804	7877	318971	661562	90432	169697	19.97

TABLE V
SUMMARY OF TRACES 1 – 3 — ALL TRAFFIC

Trace No.	% Delayed			% of connections subject to					% Later packet loss	
	Conns.	Pages	Servers	Rtm. of early cSYN	Rtm. of early fReq.	Other No Req.	Other No Rep.	Other No conn.	Client	Server
1	4.61	3.84	8.48	0.64	0.38	0.66	3.71	0.28	2.53	8.98
2	1.75	1.48	7.43	0.67	0.40	0.58	0.83	0.13	2.81	11.54
3	2.72	2.85	7.21	1.22	0.73	0.72	1.14	0.28	3.03	9.94

TABLE VI
SUMMARY OF DELAYS: TRACES 1–3 — ALL TRAFFIC

Period	75th percentile		90th percentile	
	Page	85%	Page	85%
11.30 – 12.40	8.2	6.3	18.1	14.2
Less delays	7.6	5.5	16.9	12.2
12.40 – 1.50	35.6	26.4	58.0	46.6
Less delays	25.0	12.6	37.9	25.0

TABLE IV
75TH AND 90TH PERCENTILES OF NEWS SITE PAGE DOWNLOAD TIMES IN SECONDS FOR WHOLE PAGES AND 85% OF CONTENT

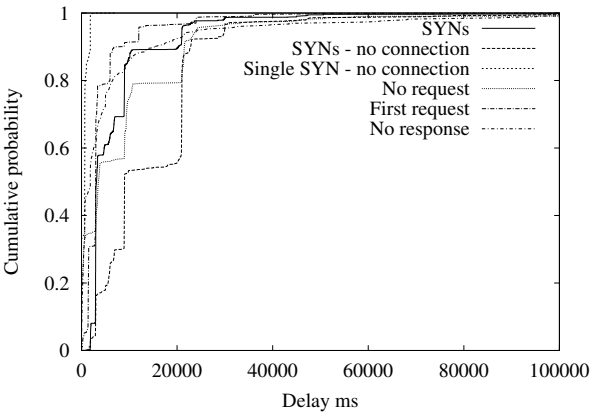

Fig. 8. The factors contributing to delay — the news site: CDF of delays contributed by retransmissions of SYNs and first requests, no request on connection or no response from connected server

same monitoring point and at the same time of day. Traffic to the news site has been excluded from the analysis upon which these tables are based.

Comparison with Tables II and III show that approximately 4.4% of connections in the general sample suffer from early loss (as opposed to approximately 14.4% in the case of the news server); similar proportions for the number of pages visited are 2.7% and 18.82%. Although the incidence of early delay is markedly less in the general sample, the potential magnitude of the delay which can be caused in page-download time may be subjectively annoying to the user in the instances where it occurs. We can only surmise that the traffic from our site may have a higher ratio of visits to well-provisioned servers than that of the general population.

VII. IMPLICATIONS FOR HTTP PERSISTENT CONNECTIONS

When non-persistent connections are used to download pages early loss on one of the concurrent connections used to fetch components does not necessarily lead to serious delay — as shown in Figure 1, progress can continue using other connections. The download only becomes completely stalled when the browser's entire 'allowance' of concurrent connections are delayed, as illustrated in Figure 3.

When *persistent* connections are employed the effects of early loss can cause a different pattern of delay. If a connection is stalled through early loss *all* requests using that connection will be delayed, but once a connection has been established no further such delays will occur.

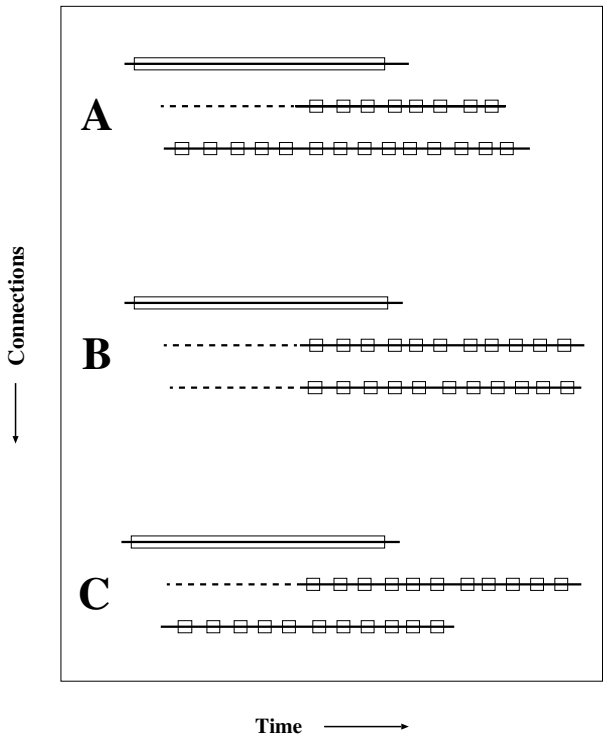

Fig. 9. Page-downloads using persistent connections. The dotted lines indicate an initial delay on the connection. In the case of page **A** one connection suffers delay while object downloads proceed on the other. All progress is initially stalled in the case of page **B**. Naïve behaviour is demonstrated by the browser downloading page **C**.

Figure 9 shows a typical pattern of page download activity using persistent connections: the container document is requested on an initial connection, and component objects are downloaded on concurrent persistent connections. In the download shown at **A** one connection is subject to delay, but activity can continue on the other; objects downloaded on the first connection, once established, are all subject to the initial delay. In the case of page download **B** both connections are initially stalled, and no progress can be made. In either case object downloads are not postponed for longer than would have been the case if one, or both, of the initial pair of non-persistent connections of a similar download had been delayed. However, once the persistent connections have been established there is no opportunity for further delay to be introduced by subsequent early loss (assuming that the connections do not remain unused for any period sufficiently long to cause a reversion to slow-start mode).

We have investigated the implications for overall page download times, and the distribution of delay, in the case of traffic to the news site had the downloads observed all been made using persistent connections. In Section III we have explained how the data collected by Nprobe from the various levels of the protocol stack enables us to accurately reconstruct page download activity. By using the detailed data available we are also able to accurately simulate the observed downloads in a variety of scenarios.

In [9] we describe a technique of modeling TCP connection activity which allows us to accurately differentiate network, end system (application), and TCP behaviour, and to quantify the contribution of each to the time taken to download individual objects. The method is based upon the data-liberation and ACK-obligation model underlying Vern Paxson's `tcpanaly` [17], but with significant differences: rather than matching observed TCP behaviour to the predefined set of characteristics of a known set of TCP implementations, our model develops a characterization of each connection from a generic base. It has the advantage of knowledge of application-level activity, and does not have to accommodate the complications introduced by packet-filter–based collection (e.g., packet loss, reordering, or duplication). Because the Nprobe monitor may be placed at any point in the network, the model does, however, have to identify packet loss both up and downstream of the monitoring point.

The *activity* model provides our simulation with precise details of network round trip times, differentiated into network transit times and application-level delay; server and browser latency; the acknowledgment, data transmission, retransmission, and slow-start behaviour of the connected hosts; and an estimation of the browser's data-sinking capacity from its advertised windows. We know the number and size of the objects comprising each page from our reconstruction of its download, the type of response from the server (e.g., whether or not an object is returned) and the relationship between objects. We also know the earliest time at which an object can be requested from the time at which the monitor observed the packet carrying the relevant link. Each page download can, therefore, be simulated in the context of the network transit time; probability of early or later packet loss; browser and server latency; and host TCP characteristics prevailing at the time.

We do, nevertheless, have to make a set of key assumptions:

- That the different pattern of data transfer would not significantly alter network characteristics (i.e., transit times and loss rates)
- That the more 'bursty' requests would not appreciably raise server latency (or alternatively that the decreased number of connections to be serviced would

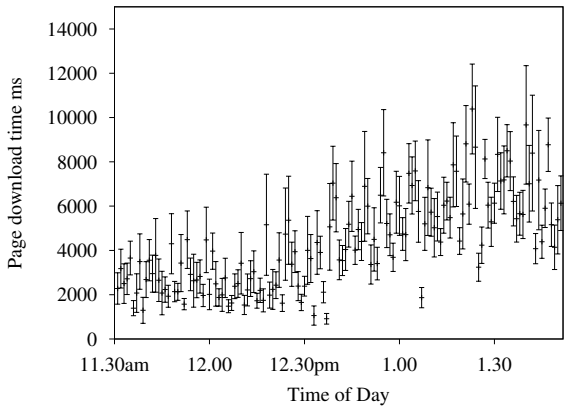

Fig. 10. Page download times for the pages downloaded from the news site using simulated persistent connections. Times are averaged over 60 second periods — the error bars show the standard error of the mean for each period

not increase the server's work rate)

- That the browser would keep pace with higher data arrival rates
- That the service latency for subsequent requests remains that of the initial request on each connection

The simulation of each page download follows the typical pattern of one connection used to request the container document, and two concurrent persistent connections upon which subsidiary components are requested, shown in Figure 9. When one of the two secondary connections are subject to initial delay, requests are channeled to the active connection until a connection is established, and thereafter each request is made on the first connection to become available. The naïve browser behaviour shown at **C** in Figure 9, in which requests are equally divided amongst connections, would result in undue delay. Our site is connected to that of the news server by a well-provisioned network, and the requirement of good connectivity stipulated in [6] would apply to the simulated connections. We have made a conservative assumption of 10 Mbps links in calculating serialization delays in the simulation's operation.

Figure 10 shows the page download times calculated by our simulation for the set of pages downloaded from the news site. Comparison with Figure 4 (which shows the actual observed times) shows that the times are considerably shorter, although they do approximately follow the same trend over the period of the trace. It is noticeable that there is a great deal more variation between averaging periods, and that shorter-term trends are much less identifiable; the pattern of activity and delay when using non-persistent connections is smoothed by the constantly shifting juxtaposition of delayed and non-delayed connections during a page download, in the case of persistent connections delay is 'all or nothing' and download times consequently vary to a much greater degree.

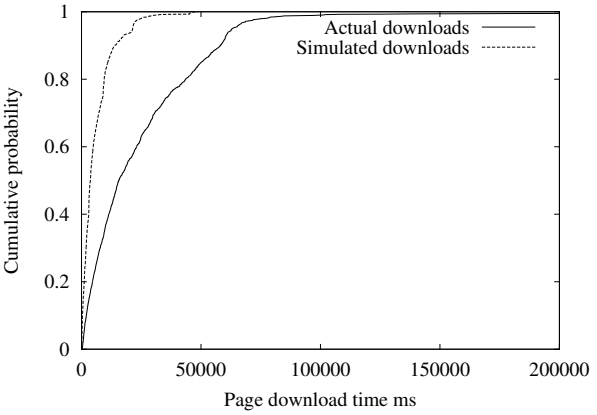

Fig. 11. The distribution of page download times for the observed and the simulated persistent connections during the second half of the news server trace

Although the simulated download times generally increase with server load over the lunch period the increase is less dramatic than in the real downloads. Figure 11 shows the relative distribution of times for both real and simulated downloads for the busy period from 12.40pm – 1.50pm. The distribution of times for the real downloads is relatively smooth, but that of the simulated connections reflects the abrupt changes in the degree of delay which are expected.

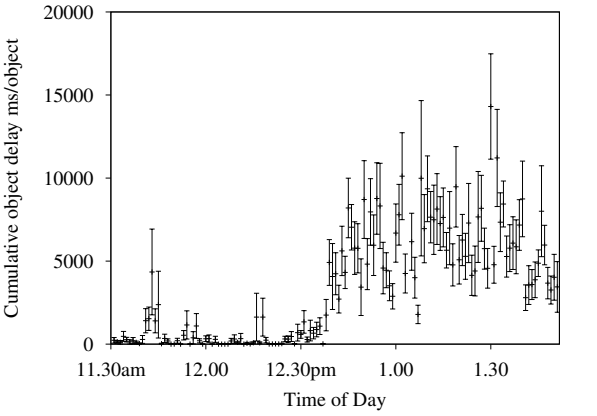

Fig. 12. *cod*s For the observed news site downloads averaged over 60 second periods

In Figures 12 and 13 we show the cumulative object delays for the real and simulated page downloads respectively, for the period of interest. The reduction in the magnitude of the per-object delay for the simulated downloads conforms with the reduction in download times, but it is interesting to note that while the average *cod* for the real traffic is approximately 25% of the average page download time, in the case of the simulated downloads this

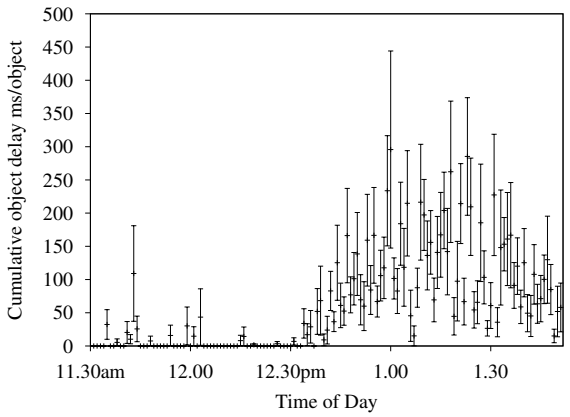

Fig. 13. *cod*s For the simulated news site downloads averaged over 60 second periods: note that this figure is not drawn to the same scale as Figure 12

proportion falls to approximately 2.5%. This difference reflects the concentration of delay at the commencement of the persistent connections, and confirms that download performance when using persistent connections is robust in the face of early loss.

VIII. Summary and Further Work

We have shown that delays in downloading Web pages must be quantified and assessed on the basis of an analysis of the whole set of TCP connections used to download their component parts, and have demonstrated a technique for doing so. We have also examined the serious delays that can be contributed to overall download times by loss early in a TCP connection's lifetime.

Our findings are based upon a traffic sample where the majority of objects were downloaded using non-persistent connections, but have implications for the use of HTTP/1.1 persistent connections: the use of fewer connections to download the components of a page minimizes the opportunity for early packet loss, but conversely, when it occurs, *all* objects downloaded on the connection will suffer a long delay. We have developing trace-based simulations which, using our detailed knowledge of page structures, prevailing network conditions, and the hosts' TCP behaviour, allow us to make detailed comparisons of performance for the observed page downloads using non-persistent and persistent connections.

Although not claiming to assess the subjective effect of page download delays we have used a simple metric to represent the distribution of delay throughout the page's component downloads, which, intuitively will impinge upon the user's perceived utility. Our calculations demonstrate that the use of persistent connections not only improves download time, but also minimizes the per-object delay during the download.

The facility to study network activity at multiple levels of the protocol stack, and to differentiate application-level effects, allows us to investigate the interactions between protocols in more detail than has generally been the case, and offers the potential for a great deal of further research. We have concentrated largely on Web traffic to date: the protocols employed are well understood and provide a useful proof of concept for our monitor, and the trace analysis tools that we are developing — but also attract further investigation made possible by the range of data that we are able to collect.

We are currently developing techniques to reliably recognize user originated download aborts, and when we have done so will investigate the correlation with delay. We have also informally noted many examples of apparently dysfunctional browser behaviour, and will establish methods of identifying and quantifying the resulting delays. The ability to gather data from HTTP headers opens the possibility of detailed study of caching behaviour. Nprobe already gathers rudimentary data from a range of other protocols, and we intend to widen our research into areas such as BGP operation and streamed media.

References

[1] T. Berners-Lee, R. Fielding, and H. Frystyk, "RFC 1945: Hypertext Transfer Protocol — HTTP/1.0," May 1996, status: INFORMATIONAL. [Online]. Available: ftp://ftp.internic.net/rfc/rfc1945.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc1945.txt

[2] V. N. Padmanabhan and J. Mogul, "Improving HTTP latency," *Computer Networks and ISDN Systems*, vol. 28(1 – 2), pp. 25 – 35, 1995.

[3] J. C. Mogul, "The case for persistent-connection http," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. ACM Press, 1995, pp. 299–313.

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "RFC 2068: Hypertext Transfer Protocol — HTTP/1.1," Jan. 1997, status: PROPOSED STANDARD (OBSOLETED by RFC 2616). [Online]. Available: ftp://ftp.internic.net/rfc/rfc2068.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc2068.txt

[5] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, and C. Lilley, "Network performance effects of http/1.1, css1, and png," in *Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM Press, 1997, pp. 155–166.

[6] J. Heidemann, K. Obraczka, and J. Touch, "Modeling the performance of http over several transport protocols," *IEEE/ACM Transactions on Networking (TON)*, vol. 5, no. 5, pp. 616–630, 1997.

[7] J. Heidemann, "Performance interactions between P-HTTP and TCP implementations," *ACM Computer Communication Review*, vol. 27(2), pp. 65–73, April 1997. [Online]. Available: citeseer.nj.nec.com/heidemann97performance.html

[8] P. Barford and M. Crovella, "Measuring web performance in the wide area," CS Department, Boston University, Tech. Rep. 1999-004, April 1999. [Online]. Available: citeseer.nj.nec.com/barford99measuring.html

[9] J. Hall, I. Pratt, and I. Leslie, "Non-intrusive estimation of web server delays," in *Proceedings of the 26th Conference on Local Computer Networks LCN2001*, Tampa, Florida, November 2001, pp. 215–224.

[10] J. Apisdorf, K. Claffy, K. Thompson, and R. Wilder., "Oc3mon: flexible, affordable, high performance statistics collection," National Laboratory for Applied Network Research., Tech. Rep., 1996, an overview is available at http://www.nlanr.net/NA/Oc3mon/.

[11] N. Anerousis, R. Caceres, N. Duffield, A. Feldmann, A. Greenberg, C. Kalmanek, P. Mishra, K. Ramakrishnan, and J. Rexford, "Using the at&t labs packetscope for internet measurements, design, and performance analysis," November 1997.

[12] J. Padhye and S. Floyd, "Identifying the TCP Behaviour of Web Servers. International Computer Science Institute, Tech. Rep. TR-01-002, February 2001.

[13] G. R. Malan and F. Jahanian, "An extensible probe architecture for network protocol performance measurement," in *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM Press, 1998, pp. 215–227.

[14] Labovitz, Malan, and Jahanian, "Internet routing instability," *IEEETNWKG: IEEE/ACM Transactions on Networking IEEE Communications Society, IEEE Computer Society and the ACM with its Special Interest Group on Data Communication (SIGCOMM), ACM Press*, vol. 6, 1998. [Online]. Available: citeseer.nj.nec.com/labovitz97internet.html

[15] A. Feldmann, "Blt: Bi-layer tracing of http and tcp/ip," in *Proceedings of the 9th International World Wide Web Conference on Computer Networks : The International Journal of Computer and Telecommunications Networking*. North-Holland Publishing Co., 2000, pp. 321–335.

[16] A. Moore, J. Hall, C. Kreibich, E. Harris, and I. Pratt, "Architecture of a Network Monitor," in *Passive & Active Measurement Workshop 2003 (PAM2003)*, Apr. 2003.

[17] V. Paxson, "Automated packet trace analysis of TCP implementations," in *Proceedings of the ACM SIGCOMM Conference : Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM-97)*, ser. Computer Communication Review, vol. 27,4. New York: ACM Press, Sept. 14–18 1997, pp. 167–180.