A Context-Aware HTML/XML Document Transmission Process For Mobile Wireless Clients[*]

Huamin Chen and Prasant Mohapatra
Department of Computer Science
Engineering II, One Shields Avenue
University of California, Davis, CA 95616.
Email: {chenhua, prasant}@cs.ucdavis.edu.

Abstract

Delivery and rendering of HTML/XML documents has been a core task in many contemporary networking applications. In mobile wireless networks, efficient handling of these types of documents is necessary due to the frequent disconnections, packet loss, and high bit error rate. One approach to address the challenge is the ability to process and reuse the partial data. Current application protocols like HTTP cannot support this approach due to the following constraints: TCP's in-order data uploading to the applications and tag matching. We propose a context-aware transmission process (CATP) to run on top of UDP. This protocol does not transmit HTML/XML files in-order. Instead, it reorganizes the files and transmit tags first before transporting their enclosed data. Conforming browsers receive the file structures and fill in with subsequent data packets in whatever sequence they arrive. As a result, lost and delayed packets do not hinder rendering of those that are logically behind but have already arrived at the client sides. Thus the retransmission of the lost frames can be concealed and overall user perceived performance improved. The user-perceivable performance is quantified in terms of silent time during which no activity is observed at the browser display. The protocol also facilitates partial content caching, amortizing network transmission overhead, and non-interactive applications of Web services. We validated this protocol through prototype implementation and compared the performance with TCP and in-order delivery UDP schemes. Our protocol provides better user-perceivable performance under various loss rates and document sizes.

Keywords*: HTTP, transmission process, UDP, partial data processing, mobile wireless devices, progressive delivery and rendering.*

[*] This research was supported in part by the National Science Foundation through the grants CCR-0296070 and ANI-0296034. A preliminary version appeared in the ICDCS Workshops, 2003.

1 Introduction

Web browsing is one of the major applications for mobile wireless devices including cell-phones, laptops, and PDAs [1]. The wireless networks are different from the wired networks in the following ways: the offered bandwidth in wireless networks is usually lower (11Mbps in IEEE 802.11b) than Ethernet networks (10-100Mbps); the data propagation latency is longer; the bit error rate is higher. As a result, mobile users often experience noticeable delays while downloading document or surfing Web sites. The delay arises either when no new data arrives to render or the received data cannot be rendered as they are out of order or their context information is incomplete. The context information in HTML/XML documents refers to the location of the nested tag pairs. Most of the present research efforts in expediting document downloading are through network protocol optimization (mostly TCP). While TCP tuning or use of other transmission processes like Streaming Control Transmission Protocol (SCTP) [37] can reduce the overall latency, they cannot help process and reuse the partial data that are either out of order or with incomplete context information.

In fact, the ability to process partial data can also help reduce the user perceivable latency such that the users can perceive a smooth data rendering in progress and the delivery of the remaining data can be concealed when the rendered data are being read. As a result, the users spend less time waiting for the data, although the overall data delivery time may be the same. Furthermore, the ability to process partial data helps reduce the memory buffer requirement to reorder the data packets. In TCP, the out-of-order packets are temporarily stored in kernel memory buffers before new packets arrive and fill the hole. Partial data with incomplete context information are stored in application level buffers before the tag matching constraint is resolved. The buffer consumption both in the kernel and application levels can be relieved if partial data can be processed, which especially benefits the devices with limited memory space.

We use *silent time* to define the time interval during which the rendering is stalled. More specifically, in the present form of browsing, silent time is the time interval between consecutive content updates. Silent time arises either because there is no packet arrival or because the packets arrive out of order. Silent time has a more significant impact on the user perceived latency than the total document download time. Thus we use the maximum silent time during a web document retrieval as the performance metric. Our overall approach to minimize the silent time is through reorganizing the retrieval process using an efficient application layer protocol.

In this paper, we proposed a context-aware transmission process (CATP) for web traffic. In CATP, the HTML/XML page contents are fragmented into several frames. The first frame (abstract frame) contains HTML/XML tags and pointers to their enclosed data. Subsequent frames (data frame) contains data that can be referenced by the pointers. The content in the data frames are self-contained in syntax and can be rendered for presentation without the help of other frames. The data frames are delivered if the contents are to the clients' interest. Thus the clients can reduce the network transmission overhead by avoiding downloading data frames that are not in their interests, which benefits the power conservation and content rendering in mobile wireless devices with small form factors. CATP also facilitates caching, content adaptation and parallel downloading from multiple servers. We have implemented the protocol in an Apache web server and evaluated its performance. The experimental results demonstrate that the silent time is significantly reduced using the proposed scheme as compared to pure UDP and TCP under various link and document characteristics.

The rest of the paper is organized as follows. Section 2 analyzes the causes of silent time during retrieval process. Section 3 discusses the transmission problems with TCP and UDP and presents CATP. Section 4 discusses the performance extensions. Section 5 presents a deployment approach. The experimental evaluation is described in Section 6, followed by the discussion on related works in Section 7. The concluding remarks are presented in Section 8.

2 Causes of the Silent Time

This section analyzes the causes of silent time during Web accesses, which are TCP's packet drop and reordering, and tag matching requirement on the rendering applications.

2.1 TCP Issues

The current web infrastructure is built on top of TCP, which provides reliable delivery between communication peers. Though TCP has been successful in delivering web traffic in many environments, previous research work has revealed the following intrinsic problems with TCP that hinders further performance improvement [27, 17]. TCP is a connection oriented protocol that ensures communication reliability. The participants first setup a communication channel by a three-way handshaking process, which is lengthy for high delay environment and short connections like HTML page delivery. Web servers could redirect the incoming requests to others because either the requested contents have moved or due to some load balancing policies employed to distribute load among multiple servers. The clients thus have to setup new connections. During the process of transmission, the receiver must send ACKs to notify successful arrivals and update the sender's TCP send window. Since the processing of ACKs is interrupt driven, large numbers of simultaneous ACKs could overwhelm the servers and erode their resources in processing more valuable tasks. If some data frames are lost during the transmission, the receivers have to wait for retransmission, which will not be launched until three duplicate ACKs successfully reach the sender or certain timeout event occurs. During this period, the receiver cannot process the out-of-order frames. As a result, the end users perceive inactivity in page rendering: a long silent time before new contents are displayed. This phenomena is called head-of-line (HOL) blocking. TCP's performance faces other challenges in the mobile wireless network. TCP congestion control and recovery mechanisms are not appropriate for wireless networks, where, instead of traffic congestion, higher bit error rate and frequent disconnections are the major causes of packet loss [5]. As a result, the congestion avoidance process, trigged by packet loss, limits the throughput.

A traffic characterization study would reveal the impact of silent time on the overall downloading latency. Though we could not find appropriate wireless traffic traces for this purpose, we discovered that, even in wired network, silent time due to packet drop and reordering is nontrivial. We investigated the maximum silent time in the Internet wide traffic from daily traces obtained from [23]. Some of the characteristics of the test-bed in [23] are as follows. The offered egress bandwidth is low (18Mbps). Most of the traffic were cross-continent and thus incurred higher transmission latency. We analyzed the difference between maximum silent time of each connection and the corresponding inter-packet arrival time. The inter-packet arrival time characterizes the network transmission latency. So the difference tells how other factors like packet loss and reordering contribute to the silent time. Out of 17,158 completed HTTP connections, 2,083 were found to have such difference. The cumulative distribution

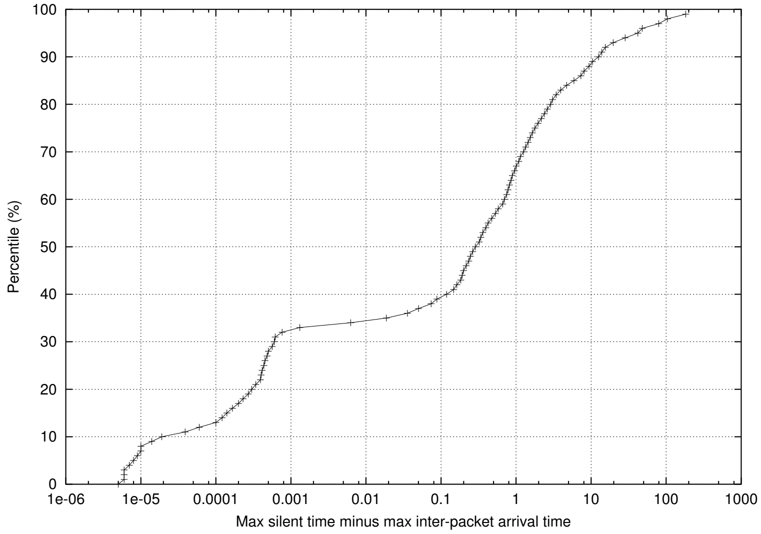

Figure 1. Cumulative distribution of maximum silent time minus maximum inter-packet arrival time.

of the difference is plotted in Figure 1. The average value is 8.6 seconds. Since the end users' tolerance latency ranges between 8 to 10 seconds [20, 21, 3], substantial requests could have aborted due to such effect. It is also found in the trace that silent time is independent of the file sizes. Thus it is likely that even during downloading small files the end users could experience lengthy silent time.

2.2 Tag Matching

The constraint of tag matching in rendering HTML pages also results in silent time. Insufficient context information, especially rendering styles, can lead to occurrence of significant silent time for users using low speed links. As reported in [29], the end users could be frustrated by long delay in retrieving large tables via dial-up connections. Early delivery of context information can resolve this problem.

Specifically, tag matching constraint arises because the start and end of the well-formed HTML tags must match to make their enclosed data meaningful. The tags are essentially predicative that establish the partial order of content within the page. Because tags can be nested, the interpretation of HTML files has to be informed of the complete context information. Consider an HTML paragraph, <H1>ABCD</H1> <H2>EFGH</H2> <H1>IJKL</H1>. Assume the paragraph is fragmented into 5 segments: <H1>ABCD, </H1><H2>, EFGH, </H2><H1>, and IJKL</H1>. Suppose these segments are transported using UDP (this is a trivial example to clarify the concept, the real UDP segments are usually quite large.), and they arrive out of order and the receiver receives segments 1, 3 and 5 first. The receiver could mistakenly think EFGH should be rendered using style H1! In this case, in order to accurately render EFGH, the receivers has to wait for the arrival of the associated tags. It is conceivable that if the tags are separated far apart, loss or delay of any data segments that are enclosed by the tags will postpone the rendering of the other segments.

We have conducted a survey of the *tag distance* in over 20 web sites including MSNBC, CNN, YAHOO, AMAZON.COM. Tag distance refers to the size of content that is enclosed by an HTML tag pairs (excluding the tags). For instance, the tag distance of HTML code <small>ABCD</small> is the string length of "ABCD" which

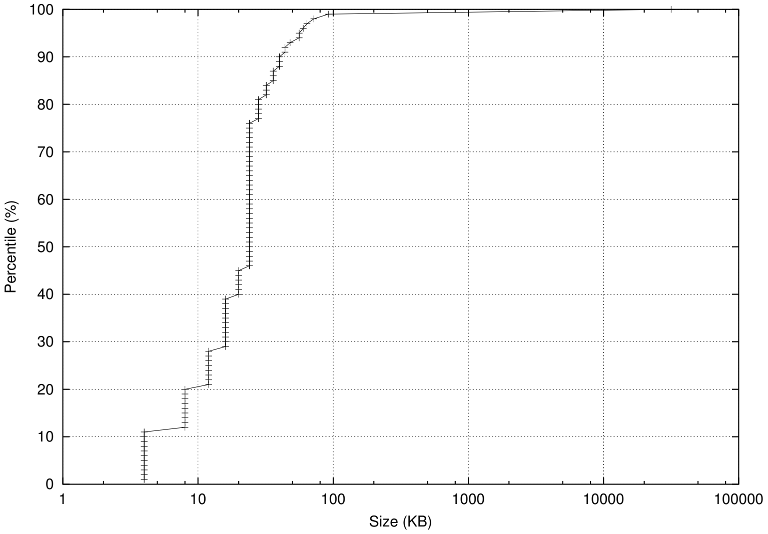

Figure 2. Cumulative distribution of file sizes.

is 4 bytes. We measured the tag distance in terms of number of the Ethernet packet size (~1500B). Obviously, tags with distance exceeding that size should be split into at least 2 packets. We only consider HTML 3.0 tags. Tags like <HTML> and <BODY> are excluded in the measurement due to their minimal impact on page presentation. Inline scripts such as JavaScript and VBScript are not counted. It is projected that, in XML documents where tags proliferate, the rendering stagnancy due to unmatched tags could be more serious.

We downloaded over 80,000 HTML files (including dynamic page outputs) using *wget*[40]. The average file size of these files is 25.5KB. The cumulative distribution of file sizes is presented in Figure 2. The cumulative distribution of the maximal tag distance in each file is plotted in Figure 3. Since x-axis is in log scale, distance 0 is omitted whose cumulative value is 55%. It is observed that, tag distance is nontrivial in the inspected pages; nearly half of these pages have to split tag pairs into several Ethernet frames and over 10% of them have tag distance more than 10 packets, which implies that browsers could have to wait for 10 packets to completely interpret the rendering style of the enclosed content! Furthermore, loss or delay of any single frames within this range could leave the rest of them unrenderable.

We believe that the average size of Web pages increases with the wide usage of authoring and content management tools for better presentation and management. The XML documents that are used in data intensive applications are expected to be even bigger. Wireless networks as well as wired low speed networks will suffer from this tendency. The bandwidth offered by wireless networks will remain low before new infrastructures are deployed. Low speed wired links like dial-up networks will continue to be a significant means of network connections (according to [19], 93% of U.S. home users are using dial-up connections, and 14.5 million new users subscribed to dial-up services versus 0.7 million to high-speed providers. Although this number changes over time, there are still a large number of dial-up users today). As a result, silent time due to transmission of large Web pages on these networks still remains an outstanding problem.

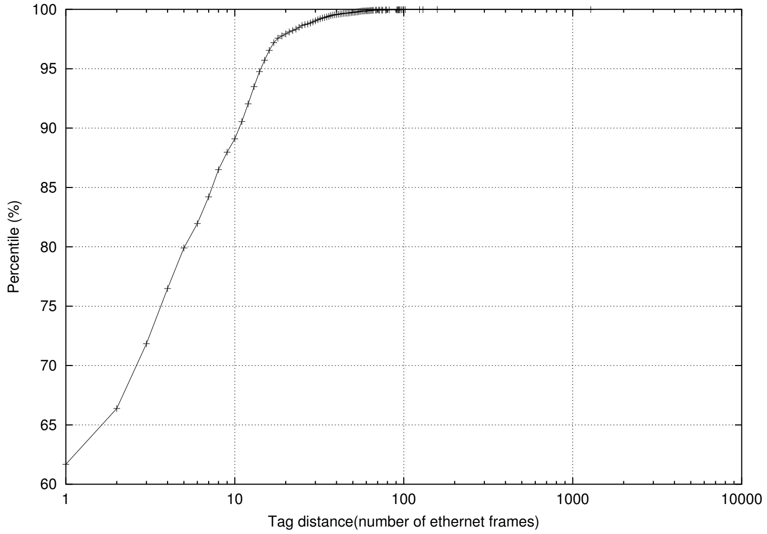

Figure 3. Cumulative distribution of tag distances.

2.3 Summary

It is inferred that if the applications are able to (1) access out-of-order packets and (2) have full knowledge of the packets' rendering context, the HOL problem can be alleviated. The first sub-problem stems from TCP's packet re-organization and the second sub-problem is due to HTML/XML documents' nested organization.

The TCP's packet re-organization is absent in UDP. Therefore we are motived to use UDP to implement our transmission process. UDP is connectionless and thus a lightweight alternative to TCP. There are no obligatory ACKs in UDP and request redirection or handover among multiple servers do not require the setup of new connections. These properties are suitable for mobility and energy conservation requirements in mobile wireless networks. There have been some research initiatives that investigate the feasibility of using UDP as the transmission process for web traffic [15, 30]. A mobile UDP protocol was also proposed in [4]. On the other hand, it is also feasible to tune the TCP protocol stack to mitigate these drawbacks.

The second problem can be resolved by the proposed context-aware transmission process as in the following sections.

3 Context-Aware Transmission Process

It is obvious that if the application can handle partial data, not only the silent time is amortized but also the packet retransmission could be concealed in the rendering process. Since TCP only delivers in-order data streams to applications, we chose UDP as our transportation protocol. The data reliability is handled by the application layer.

The aggregation of tags is the skeleton or structure of the page. Usually, when tags are stripped off, content blocks enclosed by tag pairs are rank-less. We thus propose to separate HTML tags from their enclosed content and deliver them in different packets. We call this scheme context-aware transmission process (CATP). It shares some similarities with the progressive JPEG [38] algorithm. Our scheme is illustrated and compared in the following

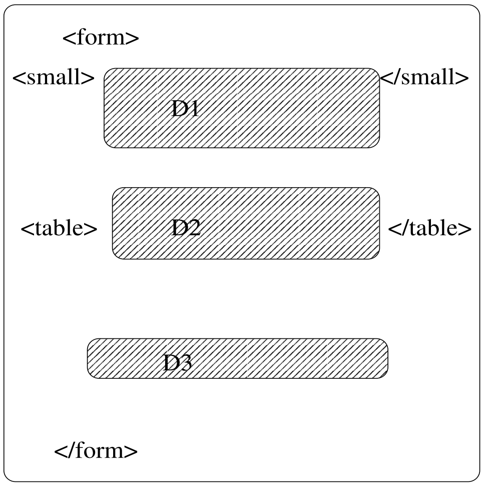

Figure 4. Sample HTML page.

diagrams.

Figure 4 illustrates an HTML page with three data sets: paragraph D1, table D2, and control section D3 that includes submission buttons. The tag pairs <form>, and </form>, <table> and </table>, <small> and </small> determines the presentation style of their enclosed data sets. Consider the following possible transmission process illustrated in Figures 5 and 6 that could be encountered in current TCP protocol and in-order UDP delivery. CATP is shown in Figure 7 where pDi is the pointer to data set Di.

In Figure 5, the rendering of received data set cannot be done until the TCP layer receives in-order data. So if fragment <table>D2 is lost during transmission, the receiver's TCP layer will not transfer the fragment D3 to the browser. The user has to wait for the retransmission of the lost segment. The silent time, denoted by T, in this occasion is determined by link loss and retransmission algorithm and if the sender uses TCP fast recovery algorithm, the silent time T is

$$\begin{aligned} T = min(P(arrival\, of\, 1\, packet) * TO, \\ P(arrival\, of\, 3\, packets) * D) \\ = min((1-p) * p^n * n * TO, \\ ((\begin{array}{c} n+3 \\ 3 \end{array}) * p^n * (1-p)^3 * (n+3)) * D), \end{aligned}$$

where p is the packet loss probability, $n \in [0, \infty)$ is the number of lost packets, TO is the retransmission timeout value, and D is the link delay. The expected silent time is

$$E(T) = min(\frac{TO}{1-p}, (\frac{3+p}{(1-p)^2}) * D).$$

It is induced that in TCP protocol, the expected silent time is determined by retransmission timeout, link delay, and packet loss probability. Larger timeout value and higher packet loss probability will extend the silent time. In

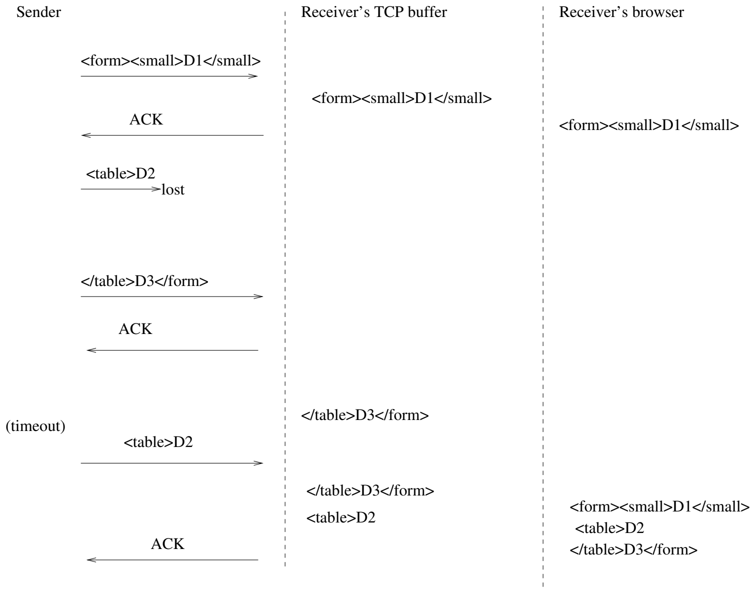

Figure 5. TCP transmission flow.

an ideal network where packet loss probability is zero and no out-of-order packet occurs, the silent time equals the link delay.

In the UDP enabled in-order delivery scheme illustrated in Figure 6, though the application can read out-of-order packets, it cannot correctly interpret the rendering style without the complete context information. Thus the loss of fragment </small><table>D2 leaves the third fragment (</table>D3</form>) with unmatched tags. The browser does not know whether the trailing tag </small> is in the lost packet or in the forthcoming packets, the rendering process cannot be done until its context information is complete. The silent time in this case is also a function of loss rate and retransmission algorithm.

In Figure 7, every transmitted packet is self-contained and independent on others. The abstract frame that contains HTML tags and pointers is sent first followed by the individual data packets. Upon arrival, the data packets fill in appropriate positions in the abstract frame. So the loss of any packet does not affect the rendering of those that have arrived successfully at the receiver side. Since the retransmission can be done in parallel with the transmission of other fragments, the silent time in this case is independent of the link loss rate and tag distance, thus the silent time due to packet loss or delay is

$$T = P(arrival\, of\, 1\, packet) * D = (1-p) * p^n * n * D,$$

and the expected silent time is

$$E(T) = \frac{D}{1-p}.$$

This formula means that the silent time incurred in context aware UDP is only related to the link delay and packet loss probability, it is not subject to out-of-order effect.

Figure 8 compares the two expected silent time functions. It is observed that the expected silent time under loss

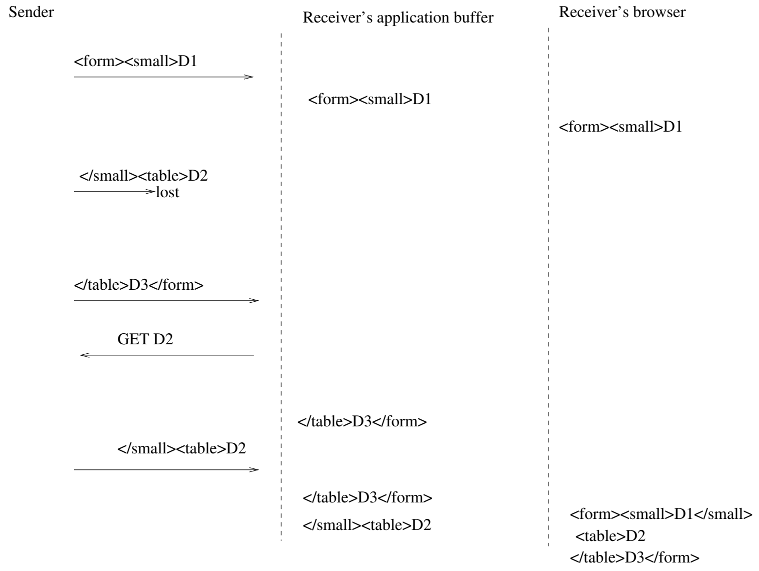

Figure 6. UDP transmission flow.

indifferent transmissions is less than the TCP transmissions.

Although the scheme in Figure 7 is not stymied by loss, it is still subject to semantical incoherence. Suppose the illustrated HTML page is a questionnaire, D1 is the direction and D2 contains the questions. If during transmission, D1 is delayed or lost but D2 arrives successfully. The end user is still confused with the rendering although he/she may not complain about the latency. In TCP, the recovery of the out-of-order packets is not activated till three duplicate ACKs are received, which is essentially context unaware. In the context aware protocol on UDP, the retransmission of logically related packets can be activated at an early stage. The scheme is illustrated in Figure 9. In this scheme, the client can detect out-of-order UDP arrival by sequence number embedded in the data frame and require the server to send related frames at an early stage if they have logical consequences on those that have already arrived.

3.1 Implications of Out-of-Order Rendering

It is inferred that the proposed scheme makes out-of-order rendering possible which is a departure from the current in-order rendering scheme. This is, however, not a significant detriment to viewing of Web pages. The organization of web pages is not always logically cohesive. For instance, when reading a news page, the user is not likely to complain about the order of news items to appear in the browser. In our experience, users usually skip most of parts of the page and go directly to their desired section. When we use a search engine, the only section of interest in the search page is the text input box and submit button. The order of retrieval of the other objects would not matter much. These two properties of the current web page content justify the feasibility of the possible out-of-order delivery. On the other hand, for a large class of non-interactive applications in Web services, out-of-order rendering poses little problem.

The successful delivery of abstract frames is the premise of restoring the original organization of the web page, even the individual frames can be rendered independently. Existing transportation layer based approaches can

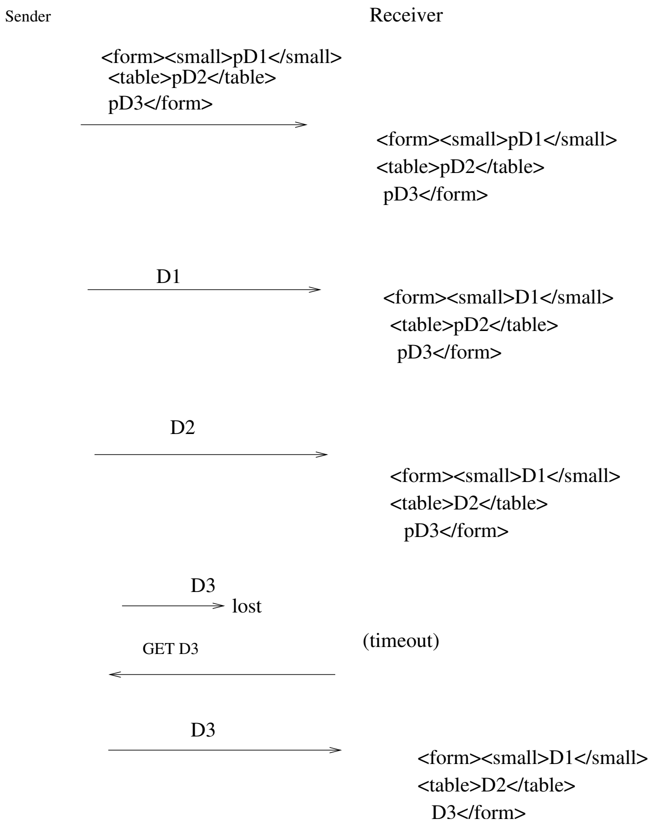

Figure 7. Loss indifferent transmission flow.

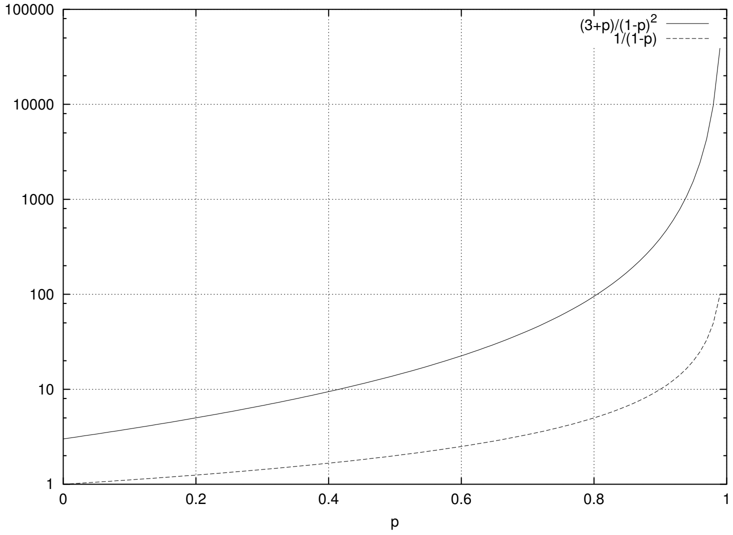

Figure 8. Expected silent time function comparison.

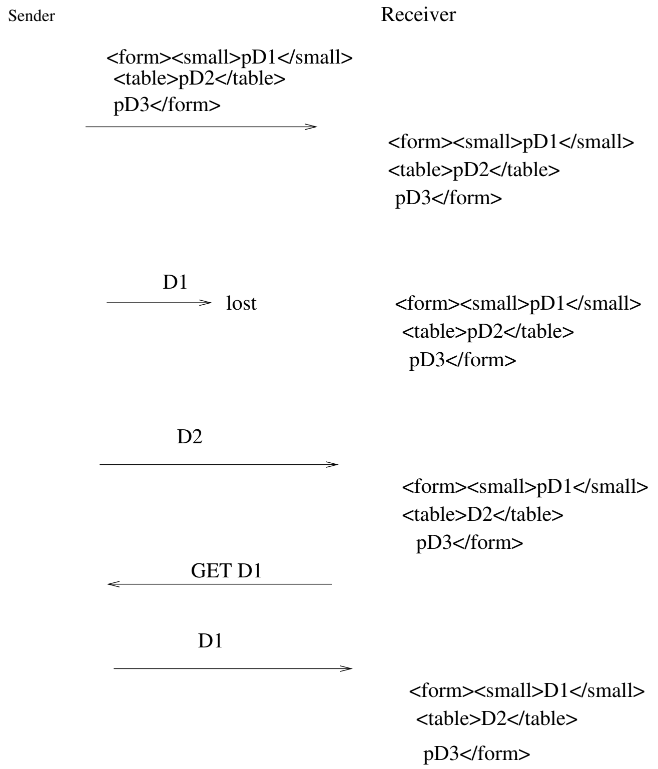

Figure 9. Out-of-order proof transmission flow.

ensure it: marking the abstract frames with high priority, using forward error correction [32], and using hybrid TCP and UDP to improve reliability. Using hybrid TCP and UDP to improve reliability has been studied in [30], where the transmission process probes the network condition and selects the best protocol to achieve high transmission rate and avoid packet loss. Reliable UDP [36] is another alternative to ensure the reliable delivery. In this study, instead of devising a new reliable transportation protocol, we utilize the application level semantics to detect and persistently request the abstract and data frames to ensure timely arrival of HTML/XML fragments. This approach is less dependent on the particular transportation protocol enhancements and is complementary with reliability assurance measures in other ISO layers.

One could argue that the HTML pages can be optimized to reduce tag distances. A simple solution is to add redundant tags that retain the presentation style. For instance, adding start and end tags in HTML paragraph <SMALL>AB</SMALL> to transform it to <SMALL>A</SMALL><SMALL>B</SMALL> essentially reduce the tag distance in half. However, this approach does not eliminate silent time due to to out-of-order arrival. Moreover, CATP does not require any redundancy in HTML tags.

4 Performance Implications

The scheme of splitting the page structure (or meta-data) from the content has many performance implications. The most notable implications are on partial content caching, and non-interactive applications.

4.1 Partial Content Caching

Since the proposed scheme splits a single Web page into grammatically complete fragments, changes of individual fragment do not affect others. Therefore caching can be more efficiently done on a fragment level. For instance, in questionnaire pages, the "instructions" are usually the same and only questions differ in each page. The server can exploit this feature by reusing the "instruction" fragments each time when a new page is requested. As a result, the retrieval process is expedited and the total traffic decreases. More significantly, the structures of Web pages are relatively stable, they can be cached locally or in proxy servers. This is especially beneficial to mobile devices.

As discussed in the preceding section, the current HTTP protocols cannot efficiently support the progressive transmission. Special mechanisms are necessary to work around such constraint. To avoid the massive overhaul of the proxies that have been deployed, it is more feasible to only adapt the proxies that serve the mobile devices who benefit most from the transmission protocol. In the future, a wide deployment of our protocol can be realized in conjunction with the fragmentation-based Web document processing [9, 8, 26] and XML technologies.

4.2 Amortizing Network Activity in Mobile Handheld Devices

Mobile handheld devices including cell-phones and PDAs have limited screen space to render the Web contents. There have been many efforts in how to efficiently render the contents with higher usability. The most prominent approaches are based on extracting the semantic data and rearranging them to fit the space limit [6] and converting the HTML document into other formats like WML (Wireless Markup Language) [39]. These approaches, however, cannot guarantee the consistency of the layout of the converted document with the original ones. As a result, people who are used to navigating by experience may be confused when switching between devices with different form factors. The progressive delivery and rendering of CATP can be applied here. The abstract frame contains sufficient layout information for the handheld devices to render. For users who are familiar with the page layout from their experiences in desktops, the navigation to their desired information can be done smoothly without requesting all the semantic information. The advantages of this approach include the retaining of the original layout and reduction of the document transmission overhead if the clients selectively request the content on-the-fly.

We have devised an interactive scheme called *scalabrowser* that exploits the preceding principle. In Scalabrowser, the server constructs the abstract frame that contains structural and some semantic information. The amount of semantic information is based on the client device's resolution and the nature of the content. Upon receiving the abstract frame, the client device renders the content and structure with sufficient legibility. The client reads the overview of the document without many scrolling efforts. When the client continues reading more of the content that is not embeded in the abstract frame, such content is retrieved through the following two strategies: it is either received during the time when the client reads the content within the abstract frame or explicitly requested on demand. In the first approach, the transmission of the remaining content is concealed. In the second approach, approach, the network activity is amortized. The detailed implementation of the interacting process and content organization and rendering algorithms are reported in [10].

4.3 Non-interactive Applications

Web crawling represents one of the non-interactive applications. When crawlers discover particular patterns of online documents using resource description framework (RDF) [31], the early awareness of their locations in the documents could reduce both the crawlers' parsing and transmission time. For instance, if the crawler has to extract the publication abstracts from library's archives, it could consume both time and network bandwidth to download and parse the whole documents which contain abstracts and other contents. While using our scheme, the crawler is enabled to read the layout of the documents from the abstract frames first and fetch the desired part without requesting the entire documents.

Similarly, an event-driven parsing application using Simple API for XML (SAX) [34] can also benefit from this scheme. Currently parsing of XML document using SAX is done in order and thus is prone to the HOL blocking. However there are usually no restrictions on the order of records or items that appear in the XML documents. If properly modified, the SAX applications can exploit the property to parse each fragment within the XML documents in parallel and independently. Figure 10 illustrates how this scheme works for the following simple XML document.

```
<book>
   <name>D1</name>
   <author>D2</author>
   <abstract>D3</abstract>
</book>
```

In this scenario, the tags are sent in the abstract frame first followed by the data frames. The modified SAX can construct the tags and their associated data and invoke proper functions to parse them. The first data frame that contains D1 is delayed in transmission and arrives at the receiver after D2 and D3. But since D2 and D3 can find their complete context information at the SAX buffer, they can be fed to the parser application unobstructed thus speeding up the parsing process.

5 Deployment of CATP

CATP is a distinct departure from current HTTP transportation in two ways: replacing TCP with UDP and out-of-order rendering. A wide range conformation takes long time to realize. We propose a deployment schedule that can exploit the performance benefit and the current infrastructure.

The use of UDP in the CATP cannot provide the same performance in congestion control and loss recovery as TCP in the Internet wide traffic. We feel there is a need for a two-stage transmission. A protocol adapter needs to be deployed between the mobile wireless clients and the Web server. The adapter is placed close to the clients in the wireless network and uses UDP for communication. The adapter interacts with the Web server using TCP over the wired Internet. The adapter's TCP is adapted to be able to relay out-of-order packets to the clients, thus avoids HOL problem. As a result, the TCP's reliability is retained and its drawbacks in the wireless network are avoided.

There are plenty of research and industrial efforts ongoing for providing wireless services through the intermediate proxies [11, 35], the functionality of CATP can be plugged into such architectures.

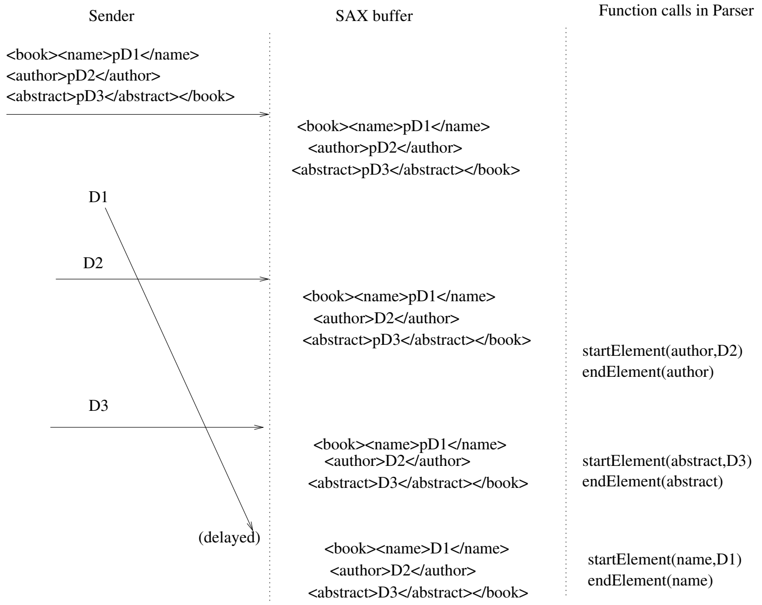

Figure 10. SAX parser for the sample XML document.

6 Experimental Evaluation

We implemented CATP in our test-bed and evaluated its performance. We used wired network to emulate different network situations. The emulation is easier to control and test different network configuration. However, due to some limitations that follow, parameters like the loss rate may not be consistent with those used in the real situations.

6.1 Implementation

CATP was implemented in Apache 1.3.22 for Linux. The modification to original Apache distribution is small, less than 50 lines of original code were modified to support UDP and a new module was added to support page fragmentation delivery. The module parses the HTTP requests and judges whether they request for the whole HTML documents or just some fragments of them. In responding to the requests for HTML fragments, it translates the fragment identifier into the file name and transmits it to the clients.

We also implemented a UDP HTTP client to benchmark the performance. In the UDP and CATP based transmission, the lost and out-of-order packets are requested persistently. More specifically, when the client is receiving HTTP packets from the server, it automatically sends requests to the server if there is no more packet arrival during certain period of time. Such period of time is set to be the round-trip time (RTT). This scheme bears certain resemblance with the NACK based transportation protocols that is commonly used to ensure reliability in multicast sessions. In the HTTP requests, the client can specify the range of the fragments that it wants to receive. The range is in the form of fragment sequence number.

The experiment was conducted on an emulated network environment with various offered link speed and loss rates by using NIST Net [22] on a Linux box with the kernel version 2.4.17. NIST Net is a Linux kernel module that emulates a WAN environment by adding time delay in packet transmission, controlling transmission rate, and constantly dropping packets. It is ideal to validate the effectiveness of our protocol in various conditions. Although NIST Net provides many network conditions, what interests us most is the variation of packet loss rates which is the major cause of HOL blocking in current Internet.

We also found some limitations in using NIST Net. During our experiment the NIST Net loss module could cause system crash if the send rate is uncontrolled, which leads to unlimited increase of queued packets. We thus added a transmission rate throttle module in our implementation that adjusts the UDP send rate below the offered link bandwidth. The presence of rate control, however, could bring performance under low lossy environment where UDP transmission cannot exploit the full offered bandwidth.

The loss algorithm in NIST Net is only based on the number of packets on the wire, regardless of the throughput. Moreover, the packet drop emulation does not emulate the link layer operations such as ARQ, the link layer's reliability assurance is diminished. As a result, the experimental results in the following sections may differ from the actual network conditions, thus they serves only for qualitative purposes and may not reflect the real situation.

We constructed two pages of 10KB and 50 KB respectively. In each of these pages, the tag distance ranges from 1 to 5 and 20 Ethernet frames, respectively. The rationale behind the page construction is that TCP's performance in retrieving small (10KB) and medium (50KB) sized documents can be monitored; the tag distance range is controllable thus its impact in silent time can be quantitatively accounted. In TCP and UDP based transmission, the silent time is measured as the maximum duration during which the out-of-order packets, if any, are re-organized. In the CATP based transmission, the silent time is maximum latency between two consecutive packet arrivals. The traffic traces were monitored by using *tcpdump* at the client side.

6.2 Experimental Results

Figures 11, 12, and 13 are the comparison of silent time of TCP, UDP and the CATP protocol under offered bandwidth 10KB/s. For each protocol, 100 requests were initiated for each loss rate configuration and their average maximum silent time is plotted.

Figure 11 represents the silent time of small tag distance. Figure 12 depicts the silent time under large tag distance and Figure 13 compares the impact of tag distance. It is observed that the large tag distance incurs longer silent time under all configurations. Note that TCP's performance is not directly affected by tag distances, but longer tag distance usually means larger file to transfer where TCP is more sensitive to loss. When the loss rate is low, TCP outperforms both UDP and our scheme due to its aggressive utilization of available bandwidth. But the advantage is very small. As the loss rate increases, TCP's silent time grows dramatically whereas our context-aware scheme remains relatively stable. Two factors are responsible for this phenomena. First, the retransmission in TCP stymies new packets arriving at the receiver thus extends the silent time. Second, upon packet loss, the TCP's congestion avoidance mechanism cuts the congestion window in half and decreases the transmission rate. These factors are absent in UDP which transmits data in constant rate. Thus the performance of UDP is better than TCP under highly lossy situations. But as discussed earlier, pure UDP does not circumvent the unmatched tag blocking thus is still subject to HOL blocking. Whereas the context-aware scheme is not obstructed by tags and thus exhibit even better performance. As demonstrated in Figure 14, the variation of tag distances has more

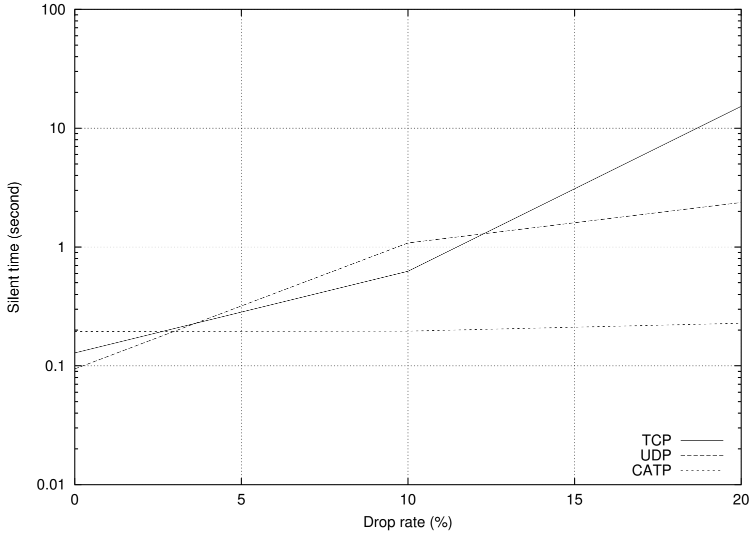

Figure 11. Tag distance is 5 frames

significant impact on pure UDP transportation. Overall CATP is better than TCP and context-unaware UDP.

For a duration of three days with 11,000 requests, an Internet wide test was conducted. There are 17 hops along the path with an average RTT of 70ms. In this test, we compared CATP and UDP protocols. The performance comparison between TCP and UDP was presented in [30]. The retransmission timeout was set as 1 second and a 10KB document size with 5 data frames was repeatedly retrieved. It was discovered that during our test most of the connections had about 130 ms silent time. A portion of connections had over 1 second silent time, they are plotted in Figure 15. It is observed that a significant number of UDP connections have more than 2 seconds silent time, while CATP connections' silent time are mostly between 1 and 2 seconds. Thus it concluded that CATP performs better than the UDP protocol in the Internet wide test.

7 Related Works

Research on HTTP transmission processes mainly focused on how to adapt TCP to transfer short HTTP files. Persistent HTTP[17, 27, 28] (P-HTTP) has been proposed to reduce the setup cost in TCP. TCP fast start [28] is a technique that utilizes prior connections' information to avoid slow start of TCP thus speed up short file downloading. Performance of various HTTP transmission processes was analyzed in [18]. The authors presented analytical models of HTTP over TCP, Asynchronized Reliable UDP Protocol (ARDP), T-TCP, and P-TCP. The major performance metrics used in these work was transmission time of the whole page. As discussed in the previous sections, this measurement may not reflect the overall perception of end users.

Using UDP to transmit Web traffic has been explored in [15, 30]. A simulation study in [15] has revealed the performance improvement under various link conditions by using UDP to transfer Web pages. A implementation-based study in [30] proposed an hybrid TCP/UDP protocol that exploits the lightweightedness and caching friendliness of UDP under low lossy links and switches to TCP otherwise. The major concerns in these studies are how to exploit the lightweight UDP to expedite the delivery of HTML pages and the extended caching benefit due to UDP's statelessness. Whereas our work only utilizes UDP's out-of-order delivery to minimize the user perceivable

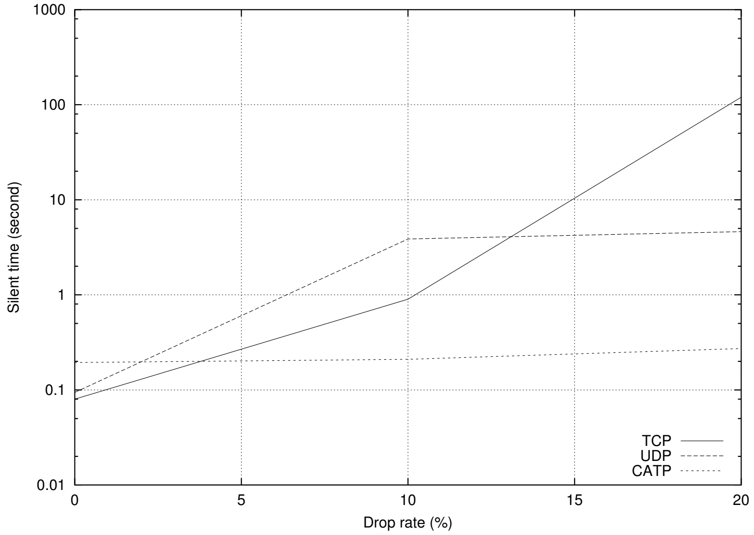

Figure 12. Tag distance is 20 frames

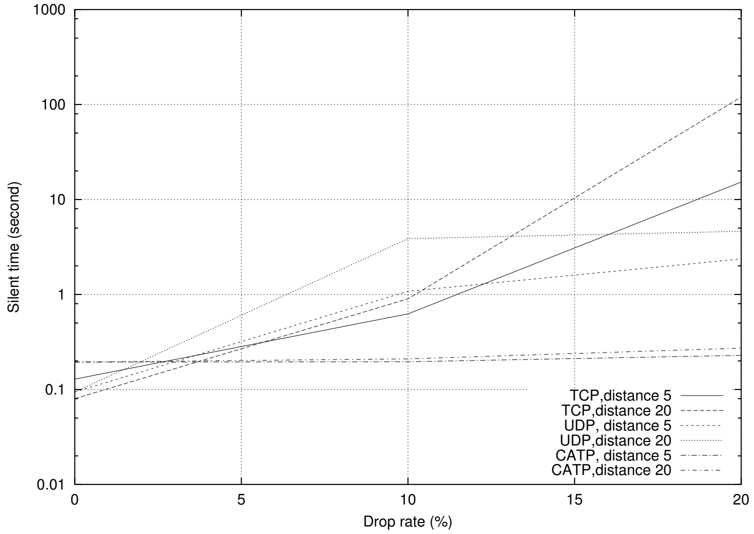

Figure 13. Tag distances are 5 and 20 frames

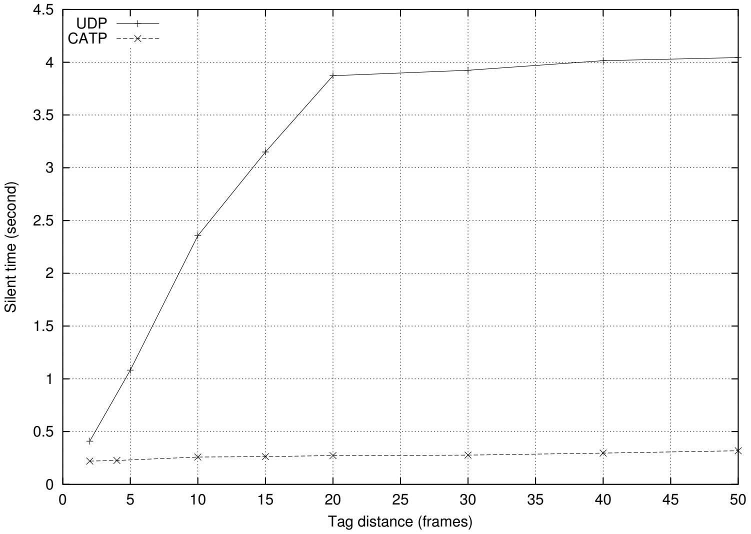

Figure 14. Silent time with various tag distances under loss rate 10%

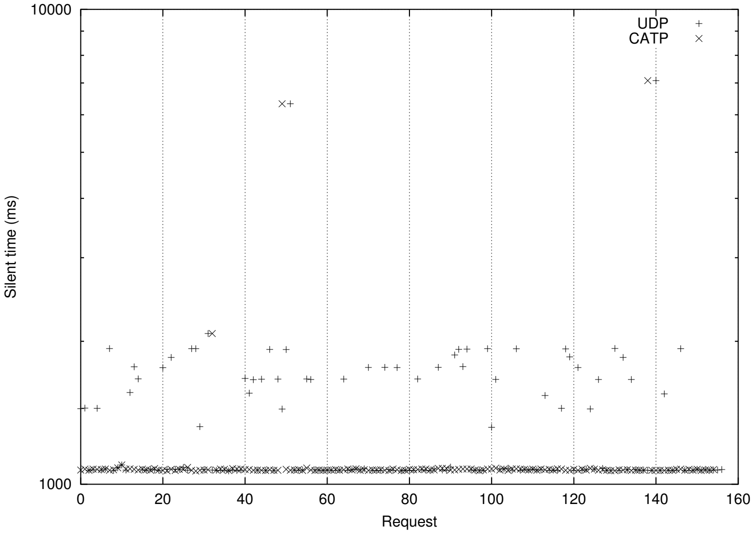

Figure 15. Internet test

silent time.

Mogul has proposed in an Internet draft [25] to eliminate the HOL blocking by adding request ID in pipelined HTTP/1.1 requests. Our scheme is applicable to both HTTP/1.0 and 1.1, at the granularity of page fragments versus requests in the draft. Parallel downloading can relieve the HOL blocking. But parallel connections to the same Web server incurs more bursty network traffic. In [33], the downloading process was conducted through multiple geographically separated servers. As a result, the traffic are distributed over multiple links. To realize the scheme, however, multiple servers should be deployed in different network domains.

There have been studies [16, 29] on how to optimize HTML pages for fast human perception. The basic means of optimization is to reduce the HTML page sizes or add extra context information. CATP is independent of page authoring, thus represents a more general approach.

Fragment-based dynamic page generation and caching have been studied in [9, 8, 26, 12, 13, 14]. Dynamic content caching based on page fragmentation was proposed in [9, 8] that uses a graphical representation of Web pages to exploit data dependency relationship. WebGraph [26] is a component-based server processing scheme to distinguish diverse attributes of each component and take appropriate measures on them. Although this paper is also based on the component-frame architecture, the way that a Web page is partitioned into various component is based on the surrounding tags and the content length. The goal of this paper is to expedite Web page transmission and rendering whereas the other works are mainly intended to facilitate dynamic page generation and caching. Web page fragmentation protocols are proposed and evaluated in [13, 14].

8 Conclusion

The head-of-line blocking in the process of Web page delivery occurs due to packet delay, loss and reordering. TCP's in oder data delivery to browsers and the long tag distance in Web pages contribute to the HOL blocking problem and results in longer user perceivable silent time. We propose a UDP-based out-of-order capable Web page transmission protocol to solve the problem. This protocol splits the structure and data in separate data frames such that the rendering of HTML data is not constrained by their enclosing tags. As a result, data blocks can be rendered immediately after their arrival at the end users without having to waiting for the complete context information. The proposed scheme amortizes the silent time and facilitates partial content caching and non-interactive applications. An implementation-based experiment validated the feasibility and demonstrated significant performance improvement under various network conditions.

References

[1] A. Adya, P. Bahl, and L. Qiu, “ Analyzing Browse Patterns of Mobile Clients,” In Proceedings of SIGCOMM Internet Measurement Workshop, San Francisco, CA, USA, November 2001.

[2] J. Bennett and C. Partridge, “Packet Reordering is Not Pathological Network Behavior,” IEEE/ACM Transactions on Networking, Vol 7, No. 6, December, 1999.

[3] N. Bhatti, A. Bouch, and A. Kuchinsky, “Integrating User-Perceived Quality into Web Server Design,” In Proceedings of the 9th International World Wide Web Conference, Amsterdam, Netherlands , May 2000.

[4] K. Brown and S. Singh, “M-UDP: UDP for mobile cellular networks,” In ACM SIGCOMM Computer Communication Review, Volume 26 , Issue 5, October 1996.

[5] K. Brown and S. Singh, “M-TCP: TCP for Mobile Cellular Networks,” In ACM SIGCOMM Computer Communication Review, Volume 27 , Issue 5, October 1997.

[6] G. Buyukkokten, O. Kaljuvee, H. Garcia-Molina, A. Paepcke, and T. Winograd, “Efficient Web Browsing on Handheld Devices using Page and Form Summerization,” ACM Transaction on Information Systems, Volume 20, Issue 1, 2002.

[7] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data”, ACM SIGCOMM 1998.

[8] J. Challenger, A. Iyengar, and P. Danzig, “A Scalable System for Consistently Caching Dynamic Web Data,” IEEE INFOCOM 1999.

[9] J. Challenger, A. Iyengar, K. Witting, C. Ferstat, and P. Reed, “A Publishing System for Efficiently Creating Dynamic Web Content,” IEEE INFOCOM 2000.

[10] H. Chen and P. Mohapatra, “A Novel Navigation and Transmission Technique for Mobile Handheld Devices,” Technical Report CSE-2003-1, Department of Computer Science, University of California at Davis. Available at `http://www.cs.ucdavis.edu/research/tech-reports/2003/CSE-2003-1.pdf`.

[11] H. Rao, Y. Chen, D. Chang, and M. Chen, “iMobile: A Proxy-based Platform for Mobile Services,” In The First ACM Workshop on Wireless Mobile Internet (WMI) 2001.

[12] F. Douglis, A. Haro, and M. Rabinovich, “HPP: HTML Macro-preprocessing to Support Dynamic Document Caching,” In USENIX Symposium on Internet Technologies and Systems, 1997.

[13] M. Rabinovich, Z. Xiao, F. Douglis, and C. Kalmanek, “Moving Edge-Side Includes to the Real Edge - the Clients,” USENIX Symposium on Internet Technologies and Systems 2003.

[14] ESI - Edge Server Include, `http://www.esi.org`.

[15] I. Cidon, R. Rom, A. Gupta, and C. Schuba, “Hybrid TCP-UDP Transport for Web Traffic,”, IEEE IPCCC 1999.

[16] Extreme HTML Optimization, http://www.webreference.com/authoring/languages/html/optimize/

[17] J. Heidemann, "Performance Interactions Between P-HTTP and TCP Implementations," ACM Computer Communication Review, vol. 27(2), pp. 65–73, April 1997.

[18] J. Heidemann, K. Obraczka, and J. Touch, "Modeling the Performance of HTTP over Several Transport Protocols, " IEEE/ACM Transactions on Networking, vol. 5, no. 5, pp. 616–630, Oct. 1997.

[19] ICONOCAST newsletter for August 17, 2000, http://www.iconocast.com/issue/20000817.html

[20] J. Nielsen, “Chapter 5, Usability Engineering.” Morgan Kaufmann, 1994.

[21] J. Nielsen, “Designing Web Usability.” New Riders Publishing; December 16, 1999.

[22] NIST NET, http://snad.ncsl.nist.gov/itg/nistnet/.

[23] MAWI Working Group Traffic Archive, http://tracer.csl.sony.co.jp/mawi/.

[24] B. A. Myers, “The Importance of Percent-Done Progress Indicators for Computer-Human Interfaces.” Proc. ACM CHI’85 Conf. (San Francisco, CA, 14-18 April), 11-17.

[25] J. Mogul, “Support for Out-of-order Responses in HTTP.” http://search.ietf.org/internet-drafts/draft-mogul-http-ooo-00.txt.

[26] P. Mohapatra and H. Chen, "WebGraph: A Framework for Managing and Improving Performance of Dynamic Web Content," Special Issue of Proxy Servers in the IEEE Journal of Selected Areas in Communications, 2002.

[27] V. N. Padmanabhan, “Addressing the Challenges of Web Data Transport.” PhD thesis, Computer Science Division, University of California at Berkeley, 1998.

[28] V. Padmanabhan and R. Katz, "TCP Fast Start: a Technique for Speeding Up Web Transfers," IEEE Globecom’98 Internet MiniConference, November 1998.

[29] PC Magazine, “Download Tables Faster,” May 7, 2002

[30] M. Rabinovich and H. Wang, “DHTTP: An Efficient and Cache-Friendly Transfer Protocol for Web Traffic.” IEEE INFOCOM 2001.

[31] Resource Description Framework (RDF), http://www.w3.org/RDF/.

[32] L. Rizzo, “Dummynet And Forward Error Correction.” In Proceedings of Freenix, June 1998.

[33] P. Rodriguez, A. Kirpal, and E. W. Biersack, “Parallel-Access for Mirror Sites in the Internet”, IEEE INFOCOM 2000.

[34] SAX: The Simple API for XML, http://www.megginson.com/SAX/.

[35] B. N. Schilit, et al., "m-Links: An Infrastructure for Very Small Internet Devices," Proc. of the 7th Annual International Conference on Mobile Computing and Networking, Rome, Italy, Jul 2001.

[36] W. R. Stevens, “Unix Network Programming–Networking APIs: Socket and XTI”, Prentice Hall, 1998.

[37] IETF, “RFC 2960: Stream Control Transmission Protocol”, http://www.ietf.org/rfc/rfc2960.txt

[38] G. Wallace, “The JPEG Still Picture Compression Standard”, Communications of the ACM, 34(4):30-44, April 1991.

[39] WAP Forum, “Wireless Markup Language,”, available at http://www.wapforum.org.

[40] Wget, http://www.gnu.org/software/wget/wget.html .

[41] WWW Consortium, http://www.w3c.org.