Measuring and Characterizing End-to-End Internet Service Performance

LUDMILA CHERKASOVA
Hewlett-Packard Laboratories
YUN FU
Duke University
WENTING TANG
Hewlett-Packard Laboratories
and
AMIN VAHDAT
Duke University

Fundamental to the design of reliable, high-performance network services is an understanding of the performance characteristics of the service as perceived by the client population as a whole. Understanding and measuring such end-to-end service performance is a challenging task. Current techniques include periodic sampling of service characteristics from strategic locations in the network and instrumenting Web pages with code that reports client-perceived latency back to a performance server. Limitations to these approaches include potentially nonrepresentative access patterns in the first case and determining the location of a performance bottleneck in the second.

This paper presents EtE monitor, a novel approach to measuring Web site performance. Our system passively collects packet traces from a server site to determine service performance characteristics. We introduce a two-pass heuristic and a statistical filtering mechanism to accurately reconstruct different client page accesses and to measure performance characteristics integrated across all client accesses. Relative to existing approaches, EtE monitor offers the following benefits: i) a latency breakdown between the network and server overhead of retrieving a Web page, ii) longitudinal information for all client accesses, not just the subset probed by a third party, iii) characteristics of accesses that are aborted by clients, iv) an understanding of the performance breakdown of accesses to dynamic, multitiered services, and v) quantification of the benefits of network and browser caches on server performance. Our initial implementation and performance analysis across three different commercial Web sites confirm the utility of our approach.

A short version of this article was published in USENIX'2002. A. Vahdat and Y. Fu are supported in part by research grant from HP and by the National Science Foundation (EIA-9972879). A. Vahdat is also supported by an NSF CAREER award (CCR-9984328).

Author's addresses: L. Cherkasova and W. Tang, Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94303; email: {lucy_cherkasova,wenting_tang}@hp.com; Y. Fu and A. Vahdat, Department of Computer Science, Duke University, Durham, NC 27708; email: {fu,vahdat}@cs.duke.edu

Categories and Subject Descriptors: C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Client/server*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*; C.4 [**Performance of Systems**]: *Measurement techniques, Modeling techniques, Design studies*; D.2.5 [**Software Engineering**]: Testing and Debugging—*Monitors*; D.2.8 [**Software Engineering**]: Metrics—*Performance measures*

General Terms: Measurement, Performance

Additional Key Words and Phrases: End-to-end service performance, network packet traces, passive monitoring, QoS, reconstruction of web page composition, web site performance

1. INTRODUCTION

Recent technology trends are increasingly leading to an environment where service, reliability, and robustness are eclipsing raw system behavior as the primary evaluation metrics for distributed services. First, the Internet is increasingly being used to deliver important services in support of business, government, education, and entertainment. At the same time, mission critical operations related to scientific instrumentation, military operations, and health services, are making increasing use of the Internet for delivering information and distributed coordination. Second, accessing a particular logical service (e.g., a news service or a bank account) typically requires the complex interaction of multiple machines and physical services (e.g., a database, an application server, a Web server, request routing, etc.) often spread across the network. Finally, the baseline performance of servers and networks continues to improve at exponential rates, often making available performance plentiful in the common case. At the same time, access to network services is inherently bursty, making order of magnitude spikes in request load relatively common.

A first step in building reliable and robust network services is tracking and understanding the performance of complex services across a diverse and rapidly changing client population. In a competitive landscape, such understanding is critical to continually evolving and engineering Internet services to match changing demand levels and client populations. By understanding current service access characteristics, sites might employ software to dynamically adapt to current network conditions, for example by reducing bandwidth overhead by transcoding Web page content, by leveraging additional replicas at appropriate locations in a content distribution network, or by reducing the data quality of query results to dynamic services, for instance, by sampling database contents.

In general, a Web page is composed of an HTML file and several embedded objects such as images. A browser retrieves a Web page by issuing a series of HTTP requests for all objects. However, HTTP does not provide any means to delimit the beginning or the end of a Web page. Since client-perceived Web server responses correspond to retrieval of Web pages, effectively measuring and analyzing the Web page download process is a critical and challenging problem in evaluating end-to-end performance.

Currently, there are two popular techniques for benchmarking the performance of Internet services. The first approach, active probing [Keynote

Systems, Inc. www.keynote.com; NetMechanic, Inc. www.netmechanics.com; Software Research Inc www.soft.com; Porivo Technologies, Inc. www.porivo.com; Gomez, Inc. www.gomez.com] uses machines from fixed points in the Internet to periodically request one or more URLs from a target Web service, record end-to-end performance characteristics, and report a time-varying summary back to the Web service. The second approach, Web page instrumentation [HP Corporation www.openview.hp.com; IBM Corporation www.tivoli.com/products/demos/twsm.html; Candle Corporation: eBusiness Assurance www.candle.com; Rajamony and Elnozahy 2001], associates code (e.g., JavaScript) with target Web pages. The code, after being downloaded into the client browser, tracks the download time for individual objects and reports performance characteristics back to the Web site.

In this paper, we present a novel approach to measuring Web site performance called EtE monitor. Our system passively collects network packet traces from the server site to enable either offline or online analysis of system performance characteristics. Using two-pass heuristics and statistical filtering mechanisms, we are able to accurately reconstruct individual page composition without parsing HTML files or obtaining out-of-band information about changing site characteristics. EtE monitor offers a number of benefits relative to existing techniques.

—Our system can determine the breakdown between the server and network overhead associated with retrieving a Web page. This information is necessary to understand where performance optimizations should be directed, for instance to improve server-side performance or to leverage existing content distribution networks (CDNs) to improve network locality. Such functionality is especially important in dynamic and personalized Web services where the CPU time for individual page access can be highly variable.

—EtE monitor tracks all accesses to Web pages for a given service. Many existing techniques are typically restricted to a few probes per hour to URLs that are pre-determined to be popular. Our approach is much more agile for changing client access patterns. What real clients are accessing determines the performance that EtE monitor evaluates.

—Given information on all client accesses, clustering techniques [Krishnamurthy and Wang 2000] can be utilized to determine network performance characteristics by network region or autonomous system. System administrators can use this information to determine which content distribution networks to partner with (depending on their points of presence) or to determine multi-homing strategies with particular ISPs. In the future, such information may be relayed back to CDNs in a cooperative environment as hints for future replica placement.

—EtE monitor captures information on page requests that are manually aborted by the client, either because of unsatisfactory Web site performance or specific client browsing patterns (e.g., clicking on a link before a page has completed the download process). Existing techniques cannot model user interactions in the case of active probing or they miss important aspects of Web site performance such as TCP connection establishment in the case of Web page instrumentation.

—Finally, EtE monitor is able to determine the actual benefits of both browser and network caches. By learning the likely composition of individual Web pages, our system can determine when certain embedded objects of a Web page are not requested and conclude that those objects were retrieved from some cache in the network.

This paper presents the architecture and implementation of our prototype EtE monitor. It also highlights the benefits of our approach through an evaluation of the performance of three different commercial Web sites using EtE monitor. Overall, we believe that detailed performance information will enable network services to dynamically adapt to changing access patterns and system characteristics to best match client QoS expectations. A key challenge to external evaluation of dynamic and personalized Web services is subjecting them to dynamic request streams that accurately reflect complex client interactions and the resulting computation across multiple tiers. While Web page instrumentation does allow evaluation under realistic access patterns, it remains difficult to break down network versus computation bottlenecks using this approach.

The delay due to the content generation process is determined by the amount of work required to generate a particular customized dynamic Web page. In a multi-tiered Web system, frequent calls to application servers and databases place a heavy load on back-end resources and may cause throughput bottlenecks and high server-side processing latency. In one of our case studies, we use EtE monitor to evaluate the performance of a Web service with highly personalized and dynamic content. There are several technical challenges for performing the analysis of such sites related to specific characteristics of dynamically generated and customized content, which we discuss in more detail in the paper. We believe that this class of Web service becomes increasingly important as more sites seek to personalize and customize their content for individual client preferences and interests. An important contribution of this work is a demonstration of the utility of our approach for comprehensive evaluation of such dynamic services.

Two main components of client-perceived response time are network transfer time and server-side processing time. The network transfer time depends on the latency and bandwidth of the underlying network connection. The server-side processing time is determined by the server hardware and the Web server technologies. Many Web sites use complex multi-tiered architectures where client requests are received by a front-tier Web server. This front tier processes client requests with the help of an application server, which may in turn access a back-end database using middleware technologies such as CORBA, RMI, and so on. Many new technologies, such as servlets [JavaServlet Technology java.sun.com/products/servlet] and Javaserver Pages [JavaServer Pages java.sun.com/products/jsp/technical.html], are popularly adopted for generating information-rich, dynamic Web pages. These new technologies and more complex Web site architectures require more complicated performance assessment of overall site design to understand their performance implications on end-user observed response time. Client-side processing overhead, such as browser rendering and cache lookup, can also affect client-perceived response times, but this of the delay is outside of the scope of our tool.

The user satisfaction with Web site response quality influences how long the user stays at the site, and determines the user's future visits. Thus, the response time observed by end users becomes a critical metric to measure and improve. Further, being able to characterize a group of clients who are responsible for a significant portion of the site's content or services as well as measuring their observed response time can help service providers make appropriate decisions for optimizing site performance.

The rest of this paper is organized as follows. In the next section, we survey existing techniques and products and discuss their merits and drawbacks. Section 3 outlines the EtE monitor architecture, with additional details in Sections 4–6. In Section 7, we present the results of three performance studies, which have been performed to test and validate EtE monitor and its approach. The studied Web sites include static, dynamic and customized Web pages. We also present specially designed experiments to validate the accuracy of EtE monitor performance measurements and its page access reconstruction power. We discuss the limitations of the proposed technique in Section 8 and present our conclusions and future work in Section 9.

2. RELATED WORK

A number of companies use active probing techniques to offer measurement and testing services including Keynote [Keynote Systems, Inc. www.keynote.com], NetMechanic [NetMechanic, Inc. www.netmechanics.com], Software Research [Software Research Inc www.soft.com], Porivo Technologies [Porivo Technologies, Inc. www.porivo.com], and Gomez [Gomez, Inc. www.gomez.com]. Their solutions are based on periodic polling of Web services using a set of geographically distributed, synthetic clients. In general, only a few pages or operations can be tested, potentially reflecting only a fraction of all users' experience. Further, active probing techniques typically cannot capture the potential benefits of browser and network caches, in some sense reflecting "worst case" performance. From another perspective, active probes come from a different set of machines than those that actually access the service. Thus, there may not always be correlation between the performance/reliability reported by the service and that experienced by end users. Finally, it is more difficult to determine the breakdown between network and server-side performance using active probing, and currently available services leveraging active probing do not provide this breakdown, making it more difficult for customers to determine where best to place their optimization efforts.

The idea of active probing is also used in tools based on browser instrumentation. e-Valid from Software Research, Inc. [Software Research Inc www.soft.com] is a well-known commercial product which provides a browser-based Web site monitoring. Page Detailer [Hellerstein et al. 1999; IBM Research www.research.ibm.com/pagedetailer] is another interesting tool from IBM Research advocating the idea of client side instrumentation. While browser/client instrumentation can capture many useful details and performance metrics about accesses from an individual instrumented client to Web pages of interest, this approach has drawbacks similar to the active probing technique: Web site performance can be assessed from a small number of instrumented clients deployed in a limited number of network locations. Typically, such browser-based tools are used for testing and debugging commercial Web sites.

Krishnamurthy et al [Krishnamurthy and Wills 2000] measured end-to-end Web performance on 9 client sites based on the PROCOW infrastructure. To investigate the effect of network latency on Web performance, a passive measurement may be required to compare the results with the application layer measurement.

Another popular approach is to embed instrumentation code with Web pages to record access times and report statistics back to the server. For instance, WTO (Web Transaction Observer) from HP OpenView suite [HP Corporation www.openview.hp.com] uses JavaScript to implement this functionality. With additional Web server instrumentation and cookie techniques, this product can record the server processing time for a request, enabling a breakdown between server and network processing time. However in general, single Web pages with non-HTML Content-Type fields, such as *application/postscript*, *application/x-tar*, *application/pdf*, or *application/zip*, cannot be instrumented. Further, this approach requires additional server-side instrumentation and dedicated resources to actively collect performance reports from clients. A number of other products and proposals [IBM Corporation www.tivoli.com/products/demos/twsm.html; Candle Corporation: eBusiness Assurance www.candle.com; Rajamony and Elnozahy 2001] employ similar techniques.

Similar to our approach, Web page instrumentation can also capture end-to-end performance information from real clients. But since the JavaScript code is downloaded to a client Web browser with the instrumented HTML file, and is executed after the page is downloaded, typically only the response time for retrieving the subsequent embedded images can be measured: it does not capture the connection establishment time and the main HTML file download time (which can be a significant portion of overall response time).

To avoid the above drawbacks, some recent work [Rajamony and Elnozahy 2001] proposes to instrument the hyperlinks for measuring the response times of the Web pages that the links point to. This technique exploits similar ideas of downloading a small amount of code written in JavaScript to a client browser when a Web page is accessed via a hyperlink. However, with this approach, the response times for pages like *index.html* (i.e. the Web pages that are accessed directly, not via links to them) cannot be measured.

There have been some earlier attempts to passively estimate the response time observed by clients from network level information. SPAND [Seshan et al. 1997; Stemm et al. 2000] determines network characteristics by making shared, passive measurements from a collection of hosts and uses this information for server selection—for routing client requests to the server with the best observed response time in a geographically distributed Web server cluster.

AT&T also has many research efforts for measuring and analyzing Web performance by monitoring the commercial AT&T IP network. Caceres et al.

[2000] describe the prototype infrastructure for passive packet monitoring on the AT&T network. Krishnamurthy et al [Krishnamurthy and Rexford 1999] discussed the importance of collecting packet-level information for analyzing Web content. In their work, they collected the information that server logs cannot provide such as packet timing, lost packets, and packet order. They discussed the challenges for Web analysis based on server logging in a related effort [Krishnamurthy and Rexford 1998].

Krishnamurthy et al [Krishnamurthy and Wills 2002] propose a set of polices for improving Web server performance measured by client-perceived Web page download latency. Based on passive server-side log analysis, they can group log entries into logical Web page accesses to classify client characteristics, which can be used to direct server adaptation. Their experiments show that even a simple classification of client connectivity can significantly improve poorly performing accesses.

The NetQoS, Inc. [NetQoS Inc. www.netqos.com] provides a tool for application performance monitoring, which exploits ideas similar to those proposed in this paper: it collects the network packet traces from server sites and reconstructs the request-response pairs (the client requests and the corresponding server responses) and estimates the response time for those pairs.

Other research work on network performance analysis includes the analysis of critical TCP transaction paths [Barford and Crovella 2000], which also decomposes network from server response time based on packet traces collected at both the server and client sides. Olshefski et al. [2001] attempt to estimate client-perceived response times at the server side and quantify the effect of SYN drops on a client response time. Meanwhile, many research efforts evaluate the performance improvements of HTTP/1.1 [Krishnamurthy and Wills 2000; Nielsen et al. 1997].

However, the client-perceived Web server responses are the retrievals of Web pages (a Web page is composed of an HTML file and several embedded objects such as images, and not just a single request-response pair). Thus, there is an orthogonal problem of grouping individual request-response pairs into the corresponding Web page accesses. EtE monitor provides the additional step of client page access reconstruction from network level packet trace aiming both to accurately assess the true end-to-end time observed by the client as well as to determine the breakdown between the server and network overhead associated with retrieving a Web page.

3. ETE MONITOR ARCHITECTURE

EtE monitor consists of four program modules shown in Figure 1:

(1) The *Network Packet Collector* module collects network packets using *tcpdump* [Tcpdump www.tcpdump.org] and records them to a *Network Trace*, enabling offline analysis.

(2) In the *Request-Response Reconstruction* module, EtE monitor reconstructs all TCP connections from the *Network Trace* and extracts HTTP transactions (a request with the corresponding response) from the payload. EtE monitor does not consider encrypted connections whose content cannot be

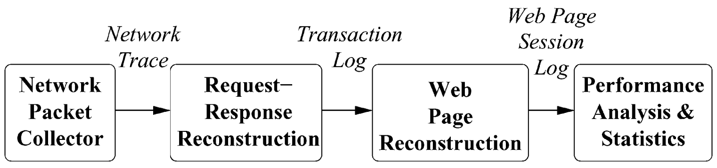


Fig. 1. EtE monitor architecture.

analyzed. After obtaining the HTTP transactions, the monitor stores some HTTP header lines and other related information in the *Transaction log* for future processing (excluding the HTTP payload). To rebuild HTTP transactions from TCP-level traces, we use a methodology proposed by Feldmann [2000] and described in more detail and extended to work with persistent HTTP connections by Krishnamurthy and Rexford [2001].

(3) The *Web Page Reconstruction* module is responsible for grouping underlying physical object retrievals together into logical Web pages (and stores them in the *Web Page Session Log*).

(4) Finally, the *Performance Analysis and Statistics* module summarizes a variety of performance characteristics integrated across all client accesses.

EtE monitor can be deployed in several different ways. First, it can be installed on a Web server as a *software component* to monitor Web transactions on a particular server. However, our software would then compete with the server for CPU cycles and I/O bandwidth (as quantified in Section 7).

Another solution is to place EtE monitor as an independent *network appliance* at a point on the network where it can capture all HTTP transactions for a Web server. If a Web site consists of multiple servers, EtE monitor should be placed at the common entrance and exit of all of them. If a Web site is supported by geographically distributed servers, such a common point may not exist. Nevertheless, distributed Web servers typically use "sticky connections": once the client has established a connection with a server, the subsequent client requests are sent to the same server. In this case, EtE monitor can still be used to capture a flow of transactions to a particular geographic site.

EtE monitor can also be configured as a *mixed solution* in which only the *Network Packet Collector* and the *Request-Response Reconstruction* module are deployed on Web servers, the other two modules can be placed on an independent node. Since the *Transaction Log* is two to three orders of magnitude smaller than the *Network Trace*, this solution reduces the performance impact on Web servers and does not introduce significant additional network traffic.

4. REQUEST-RESPONSE RECONSTRUCTION MODULE

As described above, the Request-Response Reconstruction module reconstructs all observed TCP connections. The TCP connections are rebuilt from the *Network Trace* using client IP addresses, client port numbers, and request (response) TCP sequence numbers. We chose not to use existing third-party programs to reconstruct TCP connections for efficiency. Rather than storing all connection information in the file system, our code processes and stores all information in memory for high performance. In our reconstructed TCP connections, we store all necessary IP packet-level information according to our requirements, which cannot be easily obtained from third-party software output.

Within the payload of the rebuilt TCP connections, HTTP transactions can be delimited as defined by the HTTP protocol. Meanwhile, the timestamps, sequence numbers and acknowledged sequence numbers for HTTP requests can be recorded for later matching with the corresponding HTTP responses.

When a client clicks a hypertext link to retrieve a particular Web page, the browser first establishes a TCP connection with the Web server by sending a SYN packet. If the server is ready to process the request, it accepts the connection by sending back a second SYN packet acknowledging the client's SYN.[1] At this point, the client is ready to send HTTP requests to retrieve the HTML file and all embedded objects. For each request, we are concerned with the time stamps for the first byte and the last byte of the request since they delimit the request transfer time and the beginning of server processing. We are similarly concerned with the time stamps of the beginning and the end of the corresponding HTTP response. Besides, the timestamp of the acknowledgment packet for the last byte of the response explicitly indicates that the browser has received the entire response.

EtE monitor detects aborted connections by observing either

—a RST packet sent by an HTTP client to explicitly indicate an aborted connection or
—a FIN/ACK packet sent by the client where the acknowledged sequence number is less than the observed maximum sequence number sent from the server.

After reconstructing the HTTP transactions (a request and the corresponding response), the monitor records the HTTP header lines of each request in the *Transaction Log* and discards the body of the corresponding response. Table I describes the format of an entry in the HTTP *Transaction Log*.

One alternative way to collect most of the fields of the *Transaction Log* entry is to extend Web server functionality. Apache, Netscape and IIS all have appropriate APIs. Most of the fields in the *Transaction Log* can be extracted via server instrumentation. In this case, the overall architecture of EtE monitor will be represented by the three program modules shown in Figure 2:

This approach has some merits: 1) since a Web server deals directly with request-response processing, the reconstruction of TCP connections becomes unnecessary; 2) it can handle encrypted connections.

However, the primary drawback of this approach is that Web servers must be modified, making it more difficult to deploy in the hosting center environment. Our approach is independent of any particular server technology. Additionally,

[1]Whenever EtE monitor detects a SYN packet, it considers the packet as a new connection iff it cannot find a SYN packet with the same source port number from the same IP address. A retransmitted SYN packet is not considered as a newly established connection. However, if a SYN packet is dropped, e.g. by intermediate routers, there is no way to detect the dropped SYN packet on the server side.

Table I. HTTP *Transaction Log* Entry

Field	Value
URL	The URL of the transaction
Referer	The value of the header field *Referer* if it exists
Content Type	The value of the header field *Content-Type* in the responses
Flow ID	A unique identifier to specify the TCP connection of this transaction
Source IP	The client's IP address
Request Length	The number of bytes of the HTTP request
Response Length	The number of bytes of the HTTP response
Content Length	The number of bytes of HTTP response body
Request SYN timestamp	The timestamp of the SYN packet from the client
Response SYN timestamp	The timestamp of the SYN packet from the server
Request Start Timestamp	The timestamp to receive the first byte of the HTTP request
Request End Timestamp	The timestamp to receive the last byte of the HTTP request
Response Start Timestamp	The timestamp to send the first byte of the HTTP response
Response End Timestamp	The timestamp to send the last byte of the HTTP response
ACK of Response timestamp	The ACK packet from the client for the last byte of the HTTP response
Response Status	The HTTP response status code
Via Field	Is the HTTP field *Via* is set?
Aborted	Is the TCP connection aborted?
Resent Response Packet	The number of packets resent by the server

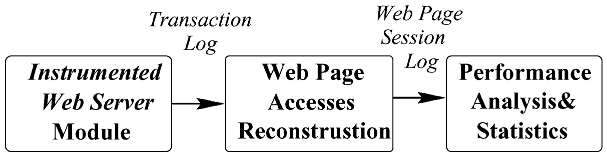

Transaction Log
Web Page Session Log
Instrumented Web Server Module
Web Page Accesses Reconstrustion
Performance Analysis& Statistics

Fig. 2. EtE monitor architecture.

EtE monitor may efficiently reflect the network level information, such as the connection setup time and resent packets, to provide complementary metrics of service performance.

5. PAGE RECONSTRUCTION MODULE

To measure the client perceived end-to-end response time for retrieving a Web page, one needs to identify the objects that are embedded in a particular Web page and to measure the response time for the client requests retrieving these embedded objects from the Web server. In other words, to measure the client perceived end-to-end response time, we must group the object requests into Web page accesses. Although we can determine some embedded objects of a Web page by parsing the HTML for the "container object," some embedded objects cannot be easily discovered through static parsing. For example, JavaScript is used in Web pages to retrieve additional objects. Without executing the JavaScript, it may be difficult to discover the identity of such objects.

Automatically determining the content of a page requires a technique to delimit individual page accesses. One recent study [Smith et al. 2001] uses an estimate of client think time as the delimiter between two pages. While this method is simple and useful, it may be inaccurate in some important cases. For example, consider the case where a client opens two Web pages from one server at the same time. Here, the requests for the two different Web pages interleave each other without any think time between them. Another case is when the interval between the requests for objects within one page may be too long to be distinguishable from think time (perhaps because of the network conditions).

As opposed to previous work, our methodology uses heuristics to determine the objects composing a Web page, the *content* of the Web page, and applies statistics to adjust the results. EtE uses the HTTP *referer* field as a major "clue" to group objects into a Web page. The *referer* field specifies the URL from which the requested URL was obtained. Thus, all requests for the embedded objects in an HTML file are recommended to set the *referer* fields to the URL of the HTML file. However, since the *referer* fields are set by client browsers, not all browsers set the fields. To solve this, EtE monitor first builds a *Knowledge Base* from those requests with *referer* fields, and uses more aggressive heuristics to group the requests without *referer* fields based on the *Knowledge Base* information.

The following simplified example shows the requests and responses that are used to retrieve the *index.html* page with the embedded image *img1.jpg* from Web server *www.hpl.hp.com*.

```
request:
  Get /index.html HTTP/1.0
  Host: www.hpl.hp.com

response:
  HTTP/1.0 200 OK
  Content-Type: text/html

request:
  Get /img1.jpg HTTP/1.0
  Host: www.hpl.hp.com
  Referer: http://www.hpl.hp.com/index.html

response:
  HTTP/1.0 200 OK
  Content-Type: image/jpeg
```

The first request is for the HTML file *index.html*. The *content-type* field in the corresponding response shows that it is an HTML file. Then, the next request is for the image *img1.jpg*. The request header field *referer* indicates that the image is embedded in *index.html*. The corresponding response shows that the content type is an image in *jpeg* format.

Subsection 5.1 outlines *Knowledge Base* construction of Web page objects. Subsection 5.2 presents the algorithm and technique to group the requests in Web page accesses using *Knowledge Base* information and a set of additional heuristics. Subsection 5.3 introduces a statistical analysis to identify valid page access patterns and to filter out incorrectly constructed accesses.

5.1 First Pass: Building a Knowledge Base of Web Page Objects

The goal of this step is to reconstruct a special subset of Web page accesses, which we use to build a *Knowledge Base* about Web pages and the objects composing them. Before grouping HTTP transactions into Web pages, EtE monitor first sorts all transactions from the *Transaction Log* using the time stamps for the beginning of the requests in increasing time order. Thus, the requests for the embedded objects of a Web page must follow the request for the corresponding HTML file of the page. When grouping objects into Web pages (here and in the next subsection), we consider only transactions with *successful* responses, that is, with status code 200 in the responses.[2]

The next step is to scan the sorted transaction log and group objects into Web page accesses. Not all the transactions are useful for the *Knowledge Base* construction process. During this step, some of the *Transaction Log* entries are excluded from our current consideration:

—Content types that are known not to contain embedded objects are excluded from the knowledge base, for example, *application/postscript*, *application/x-tar*, *application/pdf*, *application/zip* and *text/plain*. For the rest of this article, we call them *independent, single page* objects.

—If the *referer* field of a transaction is not set and its content type is not *text/html*, EtE monitor excludes it from further consideration.

To group the rest of the transactions into Web page accesses, we use the following fields from the entries in the *Transaction Log*: the request URL, the request *referer* field, the response *content type*, and the client IP address. EtE monitor stores the Web page access information into a hash table, the *Client Access Table* depicted in Figure 3, which maps a client's IP address to a *Web Page Table* containing the Web pages accessed by the client. Each entry in the *Web Page Table* is a Web page access, and is composed of the URLs of HTML files and the embedded objects. Notice that EtE monitor makes no distinction between statically and dynamically generated HTML files. We consider embedded HTML pages, for example, framed Web pages, as separate Web pages.

When processing an entry of the *Transaction Log*, EtE monitor first locates the *Web Page Table* for the client's IP in the *Client Access Table*. Then, EtE monitor handles the transaction according to its content type:

(1) If the content type is *text/html*, EtE monitor treats it as the beginning of a Web page and creates a new entry in the *Web Page Table*.

[2]In the future, we plan to extend EtE monitor to handle the requests with 304 status code. Currently, we exclude them from our consideration, because to obtain the template of a Web page it is enough to merely use the transactions with successful responses. By taking into account requests with 304 status code during the second pass, EtE monitor will be able to more accurately estimate the overall response time, especially in the case, when requests with 304 status code finish the Web page access. Since the requests with 304 status code are special "validation" transactions that do not produce responses with corresponding Web objects to transfer, they need to be handled specially to avoid skewing the performance statistics on network-related and server-side related components of response time.

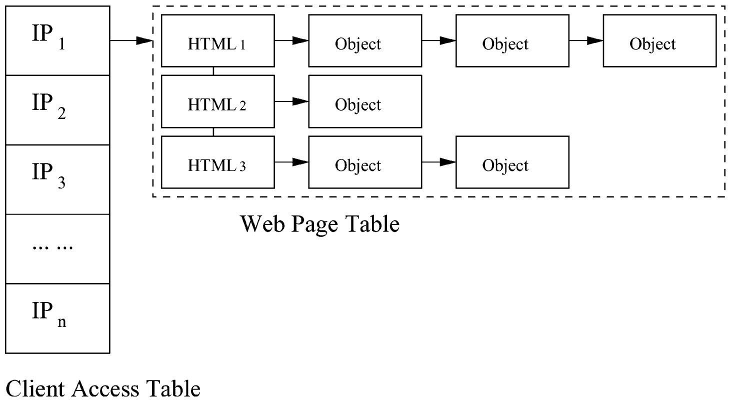


Fig. 3. Client access table.

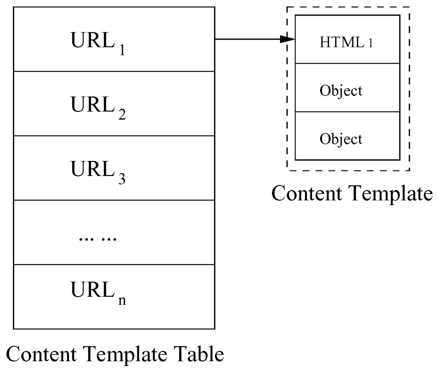


Fig. 4. *Knowledge Base* of Web pages: maps URLs to the corresponding accessed content templates.

(2) For other content types, EtE monitor attempts to insert the URL of the requested object into the Web page that contains it according to its *referer* field. If the referred HTML file is already present in the *Web Page Table*, EtE monitor appends this object at the end of the entry. If the referred HTML file does not exist in the client's *Web Page Table*, it means that the client may have retrieved a cached copy of the object from somewhere else between the client and the Web server. In this case, EtE monitor first creates a new Web page entry in the *Web Page Table* for the referred HTML file. Then it appends the considered object to this page.

From the *Client Access Table*, EtE monitor determines the *content template* of any given Web page as a combined set of all the objects that appear in all the access patterns for this page. Thus, EtE monitor scans the *Client Access Table* and creates a new hash table, as shown in Figure 4, which is used as a *Knowledge Base* to group the accesses for the same Web pages from other client's browsers that do not set the *referer* fields.

Since in this pass, the *Client Access Table* is based on an explicit reference relationship, the *Content Template Table* constructed from it is relatively trustable and can be used as a *Knowledge Base* to group the accesses for the same Web pages from other client's browsers that do not set the *referer* fields.

5.2 Second Pass: Reconstruction of Web Page Accesses

With the help of the *Knowledge Base*, EtE monitor reprocesses the entire *Transaction Log*. This time, EtE monitor does not exclude the entries without *referer* fields. It significantly extends the number of correctly processed Web page accesses. Using data structures similar to those introduced in Section 5.1, EtE monitor scans the sorted *Transaction Log* and creates a new *Client Access Table* to store all accesses as depicted in Figure 3. For each transaction, EtE monitor locates the *Web Page Table* for the client's IP in the *Client Access Table*. Then, EtE monitor handles the transaction depending on the content type:

(1) If the content type is *text/html*, EtE monitor creates a new Web page entry in the *Web Page Table*.
(2) If a transaction is an independent, single page object, EtE monitor marks it as individual page without any embedded objects and allocates a new Web page entry in the *Web Page Table*.
(3) For other content types that can be embedded in a Web page, EtE monitor attempts to insert it into the page that contains it.
    - If the *referer* field is set for this transaction, EtE monitor attempts to locate the referred page in the following way. If the referred HTML file is in an existing page entry in the *Web Page Table*, EtE monitor appends the object at the end of the entry. If the referred HTML file does not exist in the client's *Web Page Table*, EtE monitor first creates a new entry in the table for the referred page and marks it as *nonexistent*. Then it appends the object to this page. If the *referer* field is not set for this transaction, EtE monitor uses the following policies. With the help of the *Knowledge Base*, EtE monitor checks each page entry in the *Web Page Table* from the latest to earliest. If the *Knowledge Base* contains the *content template* for the checked page and the considered object does not belong to it, EtE monitor skips the entry and checks the next one until a page containing the object is found. If such an entry is found, EtE monitor appends the object to the end of the Web page.
    - If none of the entries in the *Web Page Table* contains the object based on the *Knowledge Base*, EtE monitor searches in the client's *Web Page Table* for a Web page accessed via the same flow ID as this object. If there is such a Web page, EtE monitor appends the object to the it.
    - Otherwise, if there are any accessed Web pages in the table, EtE monitor appends the object to the latest accessed one.

    If none of the above policies can be applied, EtE monitor drops the request. Obviously, the above heuristics may introduce some mistakes. Thus, EtE monitor also adopts a *configurable think time threshold* to delimit Web pages. If the time gap between the object and the tail of the Web page that it tries to append to is larger than the threshold, EtE monitor skips the considered object. In this paper, we adopt a configurable think time threshold of 4 sec.

Table II. Web Page *Probable Content Template*. There are 3075 Accesses for this Page

Index	URL	Frequency	Ratio (%)
1	/index.html	2937	95.51
2	/img1.gif	689	22.41
3	/img2.gif	641	20.85
4	/log1.gif	1	0.03
5	/log2.gif	1	0.03

5.3 Identifying Valid Accesses Using Statistical Analysis of Access Patterns

Although the above two-pass process can provide accurate Web page access reconstruction in most cases, there could still be some accesses grouped incorrectly. To filter out such accesses, we must better approximate the actual content of a Web page.

The accesses to a Web page usually exhibit various access patterns. For example, one access pattern can contain all the objects of a Web page, while other patterns may contain a subset of them (e.g., because some objects were retrieved from a browser or network caches). We assume the same access patterns of those incorrectly grouped accesses should rarely appear repeatedly. Thus, we can use the following statistical analysis on access patterns to determine the actual content of Web pages and exclude the incorrectly grouped accesses.

First, from the *Client Access Table* created in Subsection 5.2, EtE monitor collects all possible access patterns for a given Web page and identifies the *probable content template* of the Web page as the combined set of all objects that appear in all the accesses for this page. Table II shows an example of a *probable content template*. EtE monitor assigns an index for each object. The column *URL* lists the URLs of the objects that appear in the access patterns for the Web page. The column *Frequency* shows the frequency of an object in the set of all Web page accesses. In Table II, the indices are sorted by the occurrence frequencies of the objects. The column *Ratio* is the percentage of the object's accesses in the total accesses for the page.

Sometimes, a Web page may be pointed to by several URLs. For example, *http://www.hpl.hp.com* and *http://www.hpl.hp.com/index.html* both point to the same page. Before computing the statistics of the access patterns, EtE monitor attempts to merge the accesses for the same Web page with different URL expressions. EtE monitor uses the *probable content* templates of these URLs to determine whether they indicate the same page. If the *probable content* templates of two pages only differ due to the objects with small percentage of accesses (less than 1%, which means these objects might have been grouped by mistake), then EtE monitor ignores this difference and merges the URLs.

Based on the *probable content template* of a Web page, EtE monitor uses the indices of objects in the table to describe the access patterns for the Web page. Table III demonstrates a set of different access patterns for the Web page in Table II. Each row in the table is an access pattern. The column *Object Indices* shows the indices of the objects accessed in a pattern. The columns *Frequency* and *Ratio* are the number of accesses and the proportion of the pattern in the

Table III. Web Page Access Patterns

Pattern	Object Indices	Frequency	Ratio (%)
1	1	2271	73.85
2	1,2,3	475	15.45
3	1,2	113	3.67
4	1,3	76	2.47
5	2,3	51	1.66
6	2	49	1.59
7	3	38	1.24
8	1,2,4	1	0.03
9	1,3,5	1	0.03

Table IV. Web Page *True Content Template*

Index	URL
1	/index.html
2	/img1.gif
3	/img2.gif

total number of all the accesses for that page. For example, pattern 1 is a pattern in which only the object *index.html* is accessed. It is the most popular access pattern for this page: 2271 accesses out of the total 3075 accesses represent this pattern. In pattern 2, the objects *index.html*, *img1.gif* and *img2.gif* are accessed.

With the statistics of access patterns, EtE monitor further attempts to estimate the *true content template* of Web pages, which excludes the mistakenly grouped access patterns. Intuitively, the proportion of these invalid access patterns cannot be high. Thus, EtE monitor uses a configurable ratio threshold to exclude the invalid patterns (in this paper, we use 1% as a configurable ratio threshold). If the ratio of a pattern is below the threshold, EtE does not consider it as a valid pattern. In the above example, patterns 8 and 9 are not considered as valid access patterns. Only the objects found in the valid access patterns are considered as the embedded objects in a given Web page. Objects 1, 2, and 3 define the *true content template* of the Web page shown in Table IV. Based on the *true content templates*, EtE monitor filters out all the invalid accesses in a *Client Access Table*, and records the correctly constructed page accesses in the *Web Page Session Log*, which can be used to evaluate the end-to-end response performance.

6. METRICS TO MEASURE WEB SERVICE PERFORMANCE

In this section, we introduce a set of metrics and the ways to compute them in order to measure a Web service efficiency. These metrics can be categorized as:

—metrics approximating the end-to-end response time observed by the client for a Web page download. Additionally, we provide a means to calculate the breakdown between the server processing and networking portions of the overall response time.

—metrics evaluating the caching efficiency for a given Web page by computing the server file hit ratio and server byte hit ratio.

—metrics relating the end-to-end performance of aborted Web pages to the QoS.

6.1 Response Time Metrics

We use the following functions to denote the critical time stamps for connection *conn* and request r:

—$t_{syn}(conn)$: time when the first SYN packet from the client is received for establishing the connection *conn*;

—$t_{req}^{start}(r)$: time when the first byte of the request r is received ;

—$t_{req}^{end}(r)$: time when the last byte of the request r is received;

—$t_{resp}^{start}(r)$: time when the first byte of the response for r is sent;

—$t_{resp}^{end}(r)$: time when the last byte of the response for r is sent;

—$t_{resp}^{ack}(r)$: time when the ACK for the last byte of the response for r is received.

Metrics introduced in this section account for packet retransmission. However, EtE monitor cannot account for retransmissions that take place on connection establishment (i.e. due to dropped SYNs).

Additionally, for a Web page P, we have the following variables:

—N—the number of distinct connections $(conn_1, \ldots, conn_N)$ used to retrieve the objects in the Web page P;

—$r_1^k, \ldots r_{n_k}^k$—the requests for the objects retrieved through the connection $conn_k$ $(k = 1, \ldots, N)$, and ordered accordingly to the time when these requests were received, i.e.,

$$t_{req}^{end}\left(r_1^k\right) \leq t_{req}^{end}\left(r_2^k\right) \leq \cdots \leq t_{req}^{end}\left(r_{n_k}^k\right).$$

Figure 5 shows an example of a simplified scenario where a 1-object page is downloaded by the client: it shows the communication protocol for the connection setup between the client and the server as well as the set of major time stamps collected by the EtE monitor on the server side. The connection setup time measured on the server side is the time between the client SYN packet and the first byte of the client request. This represents a close approximation for the original client setup time (we present more detail on this point in subsection 7.3 when reporting our validation experiments).

If the ACK for the last byte of the client response is not delayed or lost, $t_{resp}^{ack}(r)$ is a more accurate approximation of the end-to-end response time observed by the client rather than $t_{resp}^{end}(r)$. When $t_{resp}^{ack}(r)$ is considered as the end of a transaction, it "compensates" for the latency of the first client SYN packet that is not measured on the server side. The difference between the two methods—*EtE time (last byte)* and *EtE time (ack)*—is only a round trip time, which is on the scale of milliseconds. Since the overall response time is on the scale of seconds, we consider this deviation an acceptably close approximation. To avoid the problems with delayed or lost ACKs, EtE monitor uses the time when the last byte of a response is sent by a server as the end of a transaction. Thus in the following formulae, we use $t_{resp}^{end}(r)$ to calculate the response time.

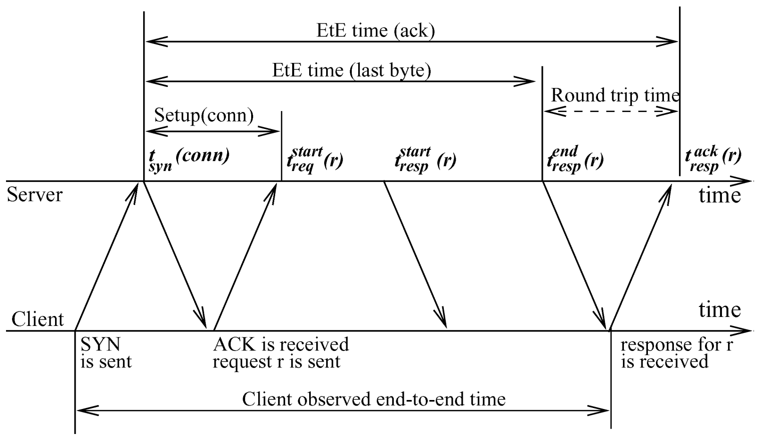


Fig. 5. An example of a 1-object page download by the client: major time stamps collected by the EtE monitor on the server side.

The extended version of HTTP 1.0 and later version HTTP 1.1 [Fielding et al. 2001] introduce the concepts of *persistent connections* and *pipelining*. Persistent connections enable reuse of a single TCP connection for multiple object retrievals from the same IP address. Pipelining allows a client to make a series of requests on a persistent connection without waiting for the previous response to complete (the server must, however, return the responses in the same order as the requests are sent).

We consider the requests $r^k_i, \ldots, r^k_n$ to belong to the same *pipelining group* (denoted as $PipeGr = \{r^k_i, \ldots, r^k_n\}$) if for any j such that $i \leq j-1 < j \leq n$, $t^{start}_{req}(r^k_j) \leq t^{end}_{resp}(r^k_{j-1})$.

Thus for all the requests on the same connection $conn_k$: $r^k_1, \ldots, r^k_{n_k}$, we define the maximum pipelining groups in such a way that they do not intersect:

$$\underbrace{r^k_1, \ldots, r^k_i}_{PipeGr_1},\ \underbrace{r^k_{i+1}}_{PipeGr_2}, \ldots,\ \underbrace{r^k_{n_k}}_{PipeGr_l}.$$

For each of the pipelining groups, we define three portions of response time: total response time (*Total*), network-related portion (*Network*), and lower-bound estimate of the server processing time (*Server*).

Let us consider the following example. For convenience, let us denote $PipeGr_1 = \{r^k_1, \ldots, r^k_i\}$.

Then

$$Total(PipeGr_1) = t^{end}_{resp}\left(r^k_i\right) - t^{start}_{req}\left(r^k_1\right),$$

$$Network(PipeGr_1) = \sum_{j=1}^{i}\left(t^{end}_{resp}\left(r^k_j\right) - t^{start}_{resp}\left(r^k_j\right)\right),$$

$$Server(PipeGr_1) = Total(PipeGr_1) - Network(PipeGr_1).$$

If no pipelining exists, a pipelining group consists of only one request. In this case, the computed server time represents precisely the server processing time for a given request-response pair.

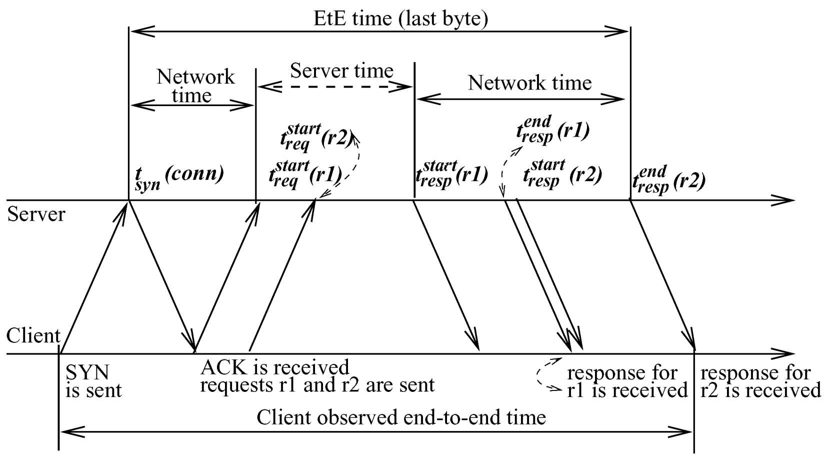


Fig. 6. An example of a pipelining group consisting of two requests, and the corresponding network-related portion and server processing portion of the overall response time.

In order to understand what information and measurements can be extracted from the time stamps observed at the server side for pipelined requests, let us consider Figure 6, which shows the communication between a client and a server, where two pipelined requests are sent in a pipelining group.

This interaction consists of: 1) the connection setup between the client and the server; 2) two subsequent requests $r1$ and $r2$ issued by the client (these requests are issued as a pipelining group); 3) the server responses for $r1$ and $r2$ are sent in the order the client requests are received by the server.

The time stamps collected at the server side reflect the time when the requests $r1$ and $r2$ are received by the server: $t_{req}^{start}(r1)$ and $t_{req}^{start}(r2)$; as well as the time when the first byte of the corresponding responses is sent by the server: $t_{resp}^{start}(r1)$ and $t_{resp}^{start}(r2)$. However, according to the HTTP 1.1 protocol, the response for $r2$ has been sent only after the response for $r1$ being sent by the server. The time between $t_{req}^{start}(r2)$ and $t_{resp}^{start}(r2)$ is indicative of the time delay on the server side before the response for $r2$ is sent to the client. However, the true server processing time for this request might be lower: the server might have processed it and simply waited for its turn to send it back to the client. The network portion of the response time for the pipelining group is defined by the sum of the network delays for the corresponding responses. This network portion of the delay defines the critical delay component in the response time.

We choose to count server processing time as only the server time that is explicitly exposed on the connection. If a connection adopts pipelining, the “real” server processing time might be larger than the computed server time because it can partially overlap the network transfer time, and it is difficult to estimate the exact server processing time from the packet-level information. However, we are still interested in estimating the “non-overlapping” server processing time as this is the portion of the server time on the critical path of overall end-to-end response time. We use this as an estimate of the lower-bound server processing time, which is explicitly exposed in the overall end-to-end response.

If connection $conn_k$ is a newly established connection to retrieve a Web page, we observe additional connection setup time:

$$Setup(conn_k) = t_{req}^{start}\left(r_1^k\right) - t_{syn}(conn_k),^3$$

otherwise the setup time is 0. Additionally, we define $t^{start}(conn_k) = t_{syn}(conn_k)$ for a newly established connection, otherwise, $t^{start}(conn_k) = t_{req}^{start}(r_1^k)$.

Similarly, we define the breakdown for a given connection $conn_k$:

$$Total(conn_k) = Setup(conn_k) + t_{resp}^{end}\left(r_{n_k}^k\right) - t_{req}^{start}\left(r_1^k\right),$$

$$Network(conn_k) = Setup(conn_k) + \sum_{j=1}^{l} Network(PipeGr_j),$$

$$Server(conn_k) = \sum_{j=1}^{l} Server(PipeGr_j).$$

Now, we define similar latencies for a given page P:

$$Total(P) = \max_{j \le N} t_{resp}^{end}\left(r_{n_j}^j\right) - \min_{j \le N} t^{start}(conn_j),$$

$$CumNetwork(P) = \sum_{j=1}^{N} Network(conn_j),$$

$$CumServer(P) = \sum_{j=1}^{N} Server(conn_j).$$

For the rest of this article, we will use the term *EtE time* interchangeably with $Total(P)$ time.

The functions $CumNetwork(P)$ and $CumServer(P)$ give the sum of all the network-related and server processing portions of the response time over all connections used to retrieve the Web page. However, the connections can be opened concurrently by the browser as shown in Figure 7, and the server processing time portion and network transfer time portion on different concurrent connections may overlap.

To evaluate the concurrency (overlap) impact, we introduce the page concurrency coefficient *ConcurrencyCoef(P)*:

$$ConcurrencyCoef(P) = \frac{\sum_{j=1}^{N} Total(conn_j)}{Total(P)}.$$

Using page concurrency coefficient, we finally compute the network-related and service-related portions of response time for a particular page P:

$$Network(P) = CumNetwork(P)/ConcurrencyCoef(P),$$

$$Server(P) = CumServer(P)/ConcurrencyCoef(P).$$

[3]The connection setup time as measured by EtE monitor does not include dropped SYNs, as discussed earlier in Section 4.

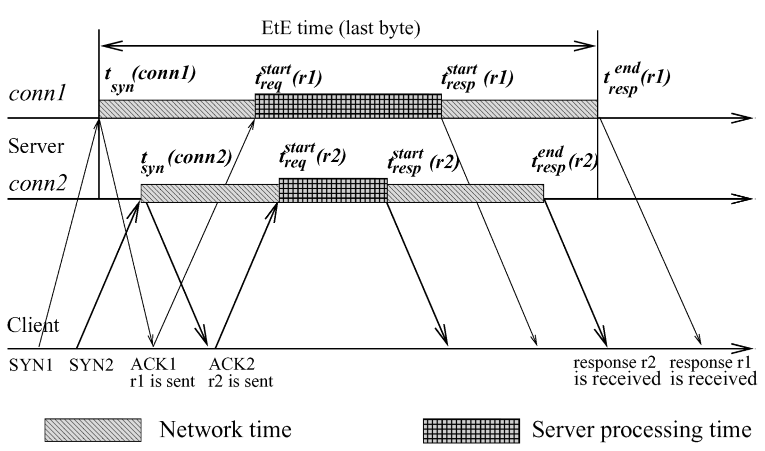


Fig. 7. An example of concurrent connections and the corresponding time stamps.

Understanding this breakdown between the network-related and server-related portions of response time is necessary for future service optimizations. It also helps to evaluate the possible impact on end-to-end response time improvements resulting from server-side optimizations.

EtE monitor can distinguish the requests sent to a Web server from clients behind proxies by checking the HTTP *via* fields. If a client page access is handled via the same proxy (which is typically the case, especially when persistent connections are used), EtE monitor provides correct measurements for end-to-end response time and other metrics, and provides interesting statistics on the percentage of client requests coming from proxies. Clearly, this percentage is an approximation, since not all the proxies set the *via* fields in their requests. Finally, EtE monitor can only measure the response time to a proxy instead of the actual client behind it.

6.2 Metrics Evaluating the Web Service Caching Efficiency

Real clients of a Web service may benefit from the presence of network and browser caches, which can significantly reduce their perceived response time. However, most existing performance measurement techniques do not provide a substantial amount of information on the impact of caches on Web services: what percentage of the files and bytes are delivered from the server compared with the total files and bytes required for delivering the Web service. This impact can only be partially evaluated from Web server logs by checking response status code 304, whose corresponding requests are sent by the network caches to validate whether the cached object has been modified. If the status code 304 is set, the cached object is not expired and need not be retrieved again.

To evaluate the caching efficiency of a Web service, we introduce two metrics: *server file hit ratio* and *server byte hit ratio* for each Web page.

For a Web page P, assume the objects composing the page are $O_1, \ldots, O_n$. Let $Size(O_i)$ denote the size of object O_i in bytes. Then we define $NumFiles(P) = n$ and $Size(P) = \sum_{j=1}^{n} Size(O_j)$.

Additionally, for each access P^i_{access} of the page P, assume the objects retrieved in the access are $O^i_1, \dots, O^i_{k_i}$, we define $NumFiles(P^i_{access}) = k_i$ and $Size(P^i_{access}) = \sum_{j=1}^{k_i} Size(O^i_j)$. First, we define *file hit ratio* and *byte hit ratio* for each page access in the following way:

$$FileHitRatio\left(P^i_{access}\right) = NumFiles\left(P^i_{access}\right)/NumFiles(P),$$

$$ByteHitRatio\left(P^i_{access}\right) = Size\left(P^i_{access}\right)/Size(P).$$

Let $P^1_{access}, \dots, P^N_{access}$ be all the accesses to the page P during the observed time interval. Then

$$ServerFileHitRatio(P) = \frac{1}{N}\sum_{k \le N} FileHitRatio\left(P^k_{access}\right),$$

$$ServerByteHitRatio(P) = \frac{1}{N}\sum_{k \le N} ByteHitRatio\left(P^k_{access}\right).$$

The lower numbers for *server file hit ratio* and *server byte hit ratio* indicate the higher caching efficiency for the Web service, that is, more files and bytes are served from network and client browser caches.

Often, a corporate Web site has a set of templates, buttons, logos, and shared images that are actively reused among a set of different pages. A user, browsing through such a site, can clearly benefit from the browser cache. The proposed caching metrics are useful for evaluating the efficiency of caching and comparing different site designs.

6.3 Aborted Pages and QoS

User-perceived QoS is another important metric to consider in EtE monitor. One way to measure the QoS of a Web service is to measure the frequency of aborted connections. The logic behind this is that if a Web site is not fast enough a user will get impatient and hit the stop button, thus aborting the connection. However, such simplistic interpretation of aborted connections and Web server QoS has several drawbacks. First, a client can interrupt HTTP transactions by clicking the browser's "stop" or "reload" button while a Web page is downloading, or clicking a displayed link before the page is completely downloaded. Thus, only a subset of aborted connections are relevant to poor Web site QoS or poor networking conditions, while other aborted connections are caused by client-specific browsing patterns. On the other hand, a Web page can be retrieved through multiple connections. A client's browser-level interruption may cause these connections to be aborted. Thus, the number of aborted page accesses more accurately reflects client satisfaction than the number of aborted connections.

For aborted pages, we distinguish the subset of pages Π_{bad} with response time higher than the given threshold X_{EtE} (in our case studies, $X_{EtE} = 6$ *sec.*). Only these pages might be reflective of the bad quality downloads. While a simple deterministic cutoff point cannot truly capture a particular client's expectation for site performance, the current industrial *ad hoc* quality goal is to deliver pages within 6 sec [Keeley 2000]. We thus attribute aborted pages that have not crossed the 6 sec threshold to individual client browsing patterns. The next step is to distinguish the reasons leading to poor response time: whether it is due to network-or server-related performance problems, or both.

7. CASE STUDIES

In this section, we present three case studies to illustrate the benefits of EtE monitor in assessing Web site performance.

—The content of the first site (*HPL Site*) is comprised of static Web pages.
—The content of the second site (*OV-Support Site*) is dynamic but without elements of content personalization.
—The third site (*IT-Support* Site) returns pages to the clients that are both dynamic and personalized.

To attract and retain customers online, many Web sites use page personalization to deliver relevant content that can be customized to enrich the user experience.

At a high level, dynamic content generation operates as follows. A user request is mapped to an invocation of a script. This script executes the necessary programs to generate the requested page. The performance of the content generation process is determined by the amount of work required to generate a particular dynamic Web page. In general, HTML pages consist of two distinct components: *content* and *layout*. Content defines the actual information comprising the page, while layout defines the page presentation: how and where the content appears on the page.

Typically, dynamic content generation may involve the following three layers during page preparation. A number of different Web technologies support these three layers. A *presentation logic layer* defines a *layout* (display) of information to users and includes formatting and transformation tasks. Presentation layer tasks are typically handled by dynamic scripts (e.g. ASP, JSP). The *business logic layer* is responsible for execution of the business logic, and is typically implemented by using component technology such as Enterprise Java Beans (EJB). The *data access layer* provides the connectivity to back-end system resources such as databases and is typically supported by standard interfaces such as JDBC or ODBC.

In multi-tiered Web systems, frequent calls to application servers and databases place a heavy load on back-end resources and may cause throughput bottlenecks and high server-side processing latency. Consider a Web site providing service to both registered users (i.e., users who have an account with the site) and non-registered users (i.e. occasional or new-coming visitors). Suppose the site allows registered users to create a user profile, which specifies the user's content preferences, for example, the choice of language for returned content. For each registered user request, the Web site retrieves the user profile preferences and generates the content according to a language choice. While for non-registered users, the Web site returns a specific default page.

An additional feature of sites with dynamic and customized content is that the user preferences are often incorporated in the requested URL via a parameter list or a client-specific cookie. Thus the requests to the same logical Web page may appear as requests to different unique URLs due to the client-specific extension or a corresponding parameter list.

Thus the most important characteristics of dynamically generated and customized content from our perspective are that:

—the requests from different users to the same logical URL may result in different page content and/or different page layout;

—the requests from different users to the same logical URL may appear as requests to different unique URLs.

From the three sites used in our case studies, only the third (*IT-Support Site*) returns both dynamic and personalized content. The content of the second site under study is represented by dynamic pages but with properties being very close to the static pages: the requests to the same URL result in the same returned page (as measured by both the content and the layout).

To demonstrate the generality of our approach to a broad range of existing network challenges and to illustrate our approach to performing accurate performance evaluation for personalized Web services, we structure the presentation of our case studies into two parts: Section 7.1 presents the measurements and analysis of the *HPL Site* and *OV-Support Site*, and Section 7.2 presents the *IT-Support Site* case study and discusses in more detail the technical challenges of the page reconstruction process and performance analysis related to the sites with both dynamic and personalized content. In the *IT-Support Site* case study, we additionally compare the EtE monitor measurements with the measurements provided by Keynote, a very popular website performance evaluation service. While EtE monitor provides detailed output of performance measurements for all three sites, we choose to include in the paper only a portion of EtE monitor measurements to demonstrate the most interesting performance results and to illustrate the utility of our newly introduced metrics. Finally, in Section 7.3, we present our validation experiments to demonstrate the correctness of EtE monitor.

7.1 HPL and OV-Support Sites' Case Study

The first site under study is the HP Labs external site (*HPL Site*), *http://www.hpl.hp.com*. Static Web pages comprise most of this site's content. We measured performance of this site for a month, from July 12, 2001 to August 11, 2001. The second site is a support site for a popular HP product family, which we call *OV-Support Site*. It uses JavaServer Pages [JavaServer Pages java.sun.com/products/jsp/technical.html] technology for dynamic page generation. The architecture of this site is based on a geographically distributed Web server cluster with Cisco Distributed Director [Cisco Distributed Director www.cisco.com] for load balancing, using "sticky connections" or "sticky sessions"—once a client has established a TCP connection with a particular Web server, the client's subsequent requests are sent to the same server. We

Table V. *At-a-Glance* Statistics for *www.hpl.hp.com* and *support* Site During the Measured Period

Metrics	HPL *url1*	HPL *url2*	OV-Support *url1*	OV-Support *url2*
EtE time	3.5 sec	3.9 sec	2.6 sec	3.3 sec
% of accesses above 6 sec	8.2%	8.3%	1.8%	2.2%
% of aborted accesses above 6 sec	1.3%	2.8%	0.1%	0.2%
% of accesses from clients-proxies	16.8%	19.8%	11.2%	11.7%
EtE time from proxies	4.2 sec	3 sec	4.5 sec	3 sec
% EtE time due to network	99.6%	99.7%	96.3%	93.5%
Page size	99 KB	60.9 KB	127 KB	100 KB
Server file hit ratio	38.5%	58%	22.9%	28.6%
Server byte hit ratio	44.5%	63.2%	52.8%	44.6%
Number of objects	4	2	32	32
Number of connections	1.6	1	6.5	9.1

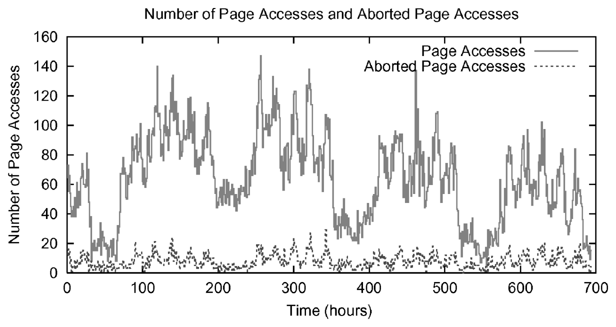
measured the site performance for 2 weeks, from October 11, 2001 to October 25, 2001. Both sites are running HTTP 1.0 servers.

Table V, called *at-a-glance*, provides the summary of the two sites' performance for the measured period using the two most frequently accessed pages at each site. The statistics in Table V are derived from the hourly statistics during the measured period.[4]

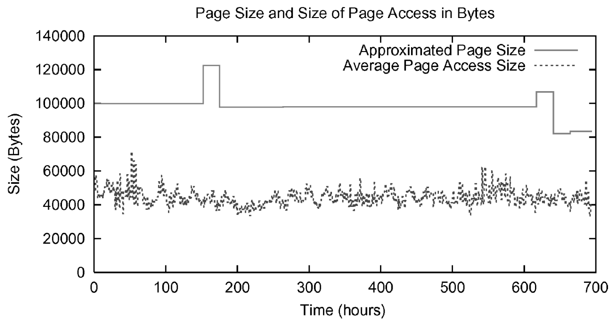
The average end-to-end response time of client accesses to these pages reflects good overall performance. However in the case of HPL, a sizeable percentage of accesses take more than 6 sec to complete (8.2%–8.3%), with a portion leading to aborted accesses (1.3%–2.8%). The OV-Support site had better overall response time with a much smaller percentage of accesses above 6 sec (1.8%–2.2%), and a correspondingly smaller percentage of accesses aborted due to high response time (0.1%–0.2%). Overall, the pages from both sites are comparable in size. However, the two pages from the HPL site have a small number of objects per page (4 and 2 correspondingly), while the OV-Support site pages are composed of 32 different objects. Page composition influences the number of client connections required to retrieve the page content. Additionally, statistics show that network and browser caches help to deliver a significant amount of page objects: in the case of the OV-Support site, only 22.9%–28.6% of the 32 objects are retrieved from the server, accounting for 44.6%–52.8% of the bytes in the requested pages. As discussed earlier, the OV-Support site content is generated using dynamic pages, which could potentially lead to a higher ratio of server processing time in the overall response time. But in general, the network transfer time dominates the performance for both sites, ranging from 93.5% for the OV-Support site to 99.7% for the HPL site.

Given the above summary, we now present more detailed information from our site measurements. For the HPL site, the two most popular pages during the observed period were *index.html* and a page in the news section describing the Itanium chip (we call it *itanium.html*).

[4]10%–15% of requests with 304 status code were excluded by EtE monitor from consideration. Most of them occur in the "middle" of the Web page accesses, and hence, they do not have significant impact on the accuracy of EtE monitor measurements.



(a)



(b)

Fig. 8. HPL site during a month: (a) Number of all and aborted accesses to *index.html*; (b) Approximated page size and average access size to *index.html*.

Figure 8(a) shows the number of page accesses to *index.html*, as well as the number of aborted page accesses during the measured period. The graph clearly reflects weekly access patterns to the site.

Figure 8(b) reflects the *approximate page size*, as reconstructed by EtE monitor—the page size of the corresponding page content template as reconstructed by EtE monitor. We use this data to additionally validate the page reconstruction process. While debugging the tool, we manually compare the content of the 20 most frequently accessed pages reconstructed by EtE monitor against the actual Web pages: the EtE monitor page reconstruction accuracy for popular pages is very high, practically 100%. Figure 8(b) allows us to "see" the results of this reconstruction process over the period of the study. In the beginning, it is a straight line exactly coinciding with the actual page size. This means that EtE monitor has reconstructed the page content template precisely. At hour mark 153, it jumps and returns to a straight line interval at the 175 hour mark. As we verified, the page was partially modified during this time interval. The EtE monitor "picked" both the old and the modified page images, since they both occurred during the same day interval and represented a significant fraction of accesses. Thus, the content template, reconstructed by EtE monitor during this time interval, has a union of images from the "old" and "new" (modified) page. However, the next day, the *Knowledge Base* was updated with correct, up-to-date page information. The second "jump" of this line corresponds to the next modification of the page. The gap can be tightened, depending on the time interval EtE monitor is set to process. The other line in

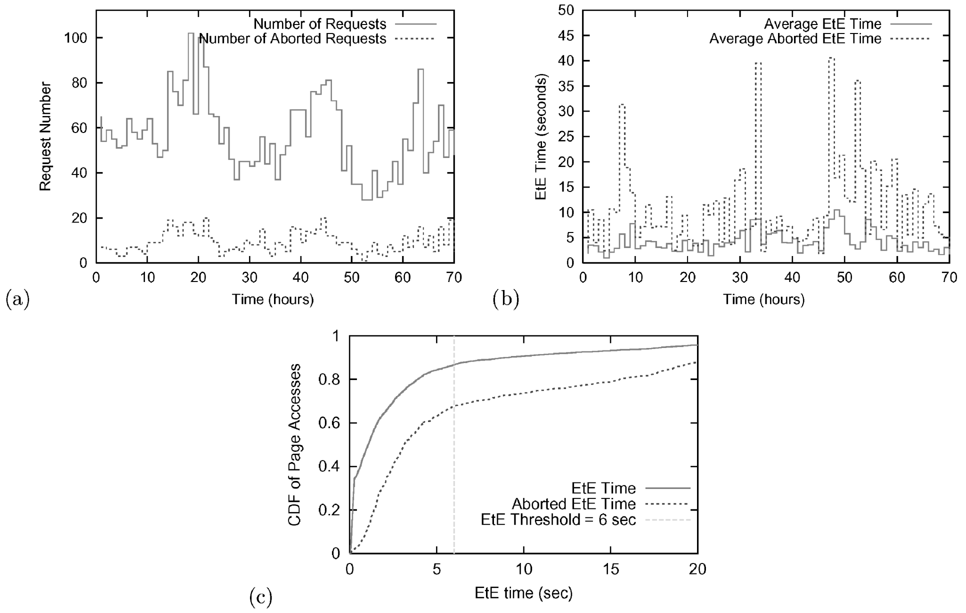


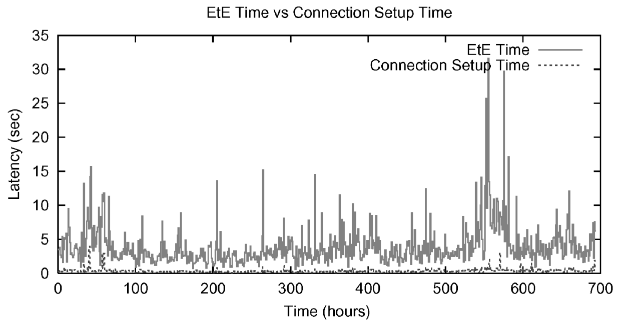
Fig. 9. HPL site during 3 days: (a) Number of all and aborted accesses to *index.html*; (b) End-to-end response times for accesses to *index.html*; (c) CDF of all and aborted accesses to *index.html* sorted by the response time in increasing order.

Figure 8(b) shows the average page access size, reflecting the server byte hit ratio of approximately 44%.

To characterize the reasons leading to the aborted Web pages, we present analysis of the aborted accesses to *index.html* page for 3 days in August (since the monthly graph looks very "busy" on an hourly scale). Figure 9(a) shows the number of all the requests and the aborted requests to *index.html* page during this interval. The number of aborted accesses (662) accounts for 16.4% of the total number of requests (4028).

Figure 9(b) shows the average end-to-end response time measured by EtE monitor for *index.html* and the average end-to-end response time for the aborted accesses to *index.html* on an hourly scale. The end-to-end response time for *index.html* page, averaged across all the page accesses, is 3.978 sec, while the average end-to-end response time of the aborted page accesses is 9.21 sec.

Figure 9(c) shows a cumulative distribution of all accesses and aborted accesses to *index.html* sorted by the end-to-end response time in increasing order. The vertical line on the graph shows the threshold of 6 sec that corresponds to an acceptable end-to-end response time. Figure 9(c) shows that 68% of the aborted accesses demonstrate end-to-end response times below 6 sec. This means that only 32% of all the aborted accesses, which in turn account for 5% of all accesses to the page, observe high end-to-end response time. The next step is to distinguish the reasons leading to a poor response time: whether it is due to network or server performance problems, or both. For all the aborted pages with high response time, the network portion dominates the overall response time

(a)


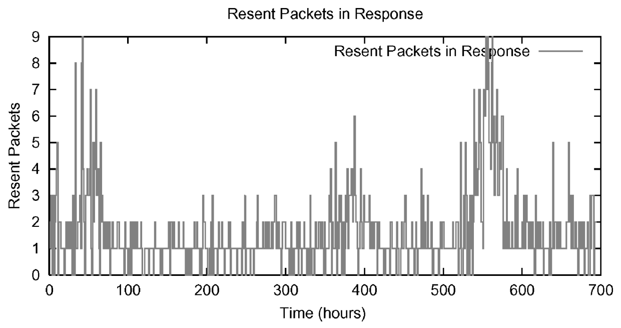
(b)


Fig. 10. HPL site during a month: (a) end-to-end response time and connection setup time for accesses to *index.html*; (b) number of resent packets in response.

(98%–99% of the total). Thus, we can conclude that performance problems are likely not server-related but rather due to congestion in the network (though it is unclear whether the congestion is at the edge or the core of the network).

Figure 10(a) shows the end-to-end response time and connection setup time for accesses to *index.html* on an hourly scale during a month. In spite of good average response time reported in at-a-glance table, hourly averages reflect significant variation in response times. The observed response time may be affected by many factors [Cardwell et al. 2000] such as network round trip time and packet loss rate, and so on. This graph helps to stress the advantages of EtE monitor and reflects the shortcomings of active probing techniques that measure page performance only a few times per hour: the collected test numbers could vary significantly from a site's instantaneous performance characteristics.

Figure 10(b) shows the number of resent packets in the response stream to clients. There are three pronounced "humps" with an increased number of resent packets. Typically, resent packets reflect network congestion or the existence of some network-related bottlenecks. Interestingly enough, such periods correspond to weekends when the overall traffic is one order of magnitude lower than weekdays (as reflected in Figure 8(a)). The explanation for this phenomenon is that during weekends the client population of the site "changes" significantly: most of the clients access the site from home using modems or other low-bandwidth connections. This leads to a higher observed end-to-end response time and an increase in the number of resent packets (i.e., TCP is likely to cause drops more often when probing for the appropriate congestion window

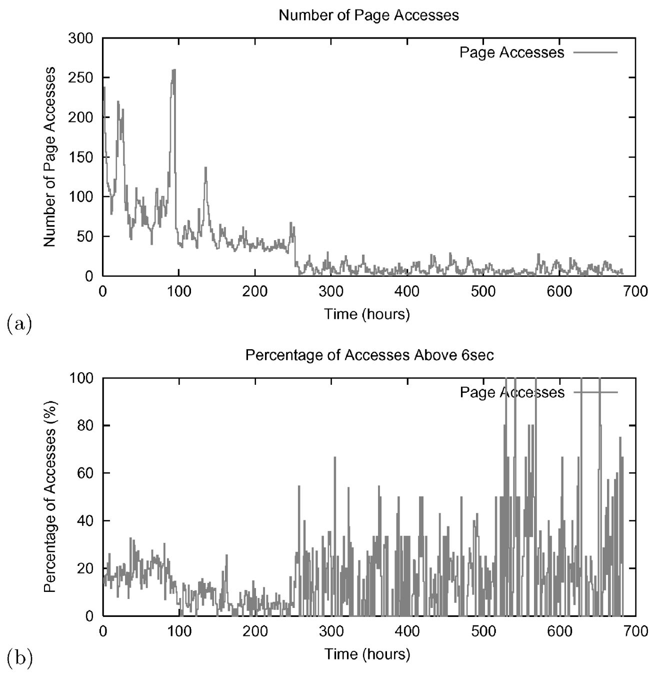


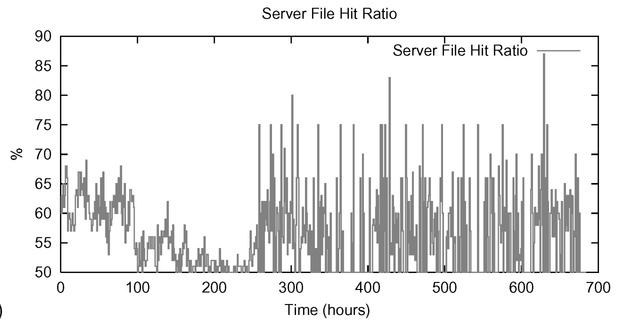
Fig. 11. HPL site during a month: (a) number of all accesses to *itanium.html*; (b) percentage of accesses with end-to-end response time above 6 sec.

over a low-bandwidth link). These results again stress the unique capabilities of EtE monitor to extract appropriate information from network packets, and reflect another shortcoming of active probing techniques that use a fixed number of artificial clients with rather good network connections to the Internet. For site designers, it is important to understand the actual client population and their end-to-end response time and the "quality" of the response. For instance, when a large population of clients has limited bandwidth parameters, the site designers should consider making the pages and their objects "lighter weight."

Figure 11(a) shows the number of page accesses to *itanium.html*. When we started our measurement of the HPL site, the *itanium.html* page was the most popular page, "beating" the popularity of the main *index.html* page. However, ten days later, this news article started to get "colder," and the page got to the seventh place by popularity.

Figure 11(b) shows the percentage of accesses with end-to-end response time above 6 sec. The percentage of high response time jumps significantly when the page becomes "colder." The reason behind this phenomenon is shown in Figure 12, which plots the server file hit and byte hit ratio. When the page became less popular, the number of objects and the corresponding bytes retrieved from the server increased significantly. This reflects the fact that fewer network caches store the objects as the page becomes less popular, forcing clients to retrieve them from the origin server.

Figure 11(b) and Figure 12 explicitly demonstrate the network caching impact on end-to-end response time. When the caching efficiency of a page is



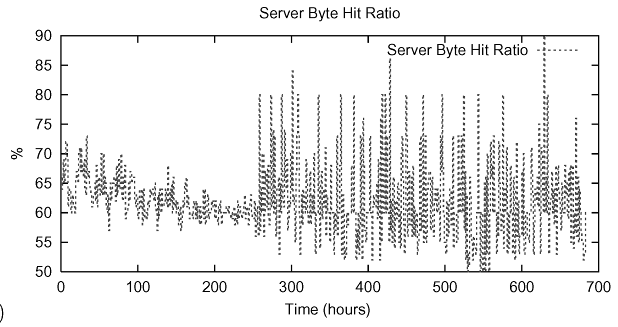


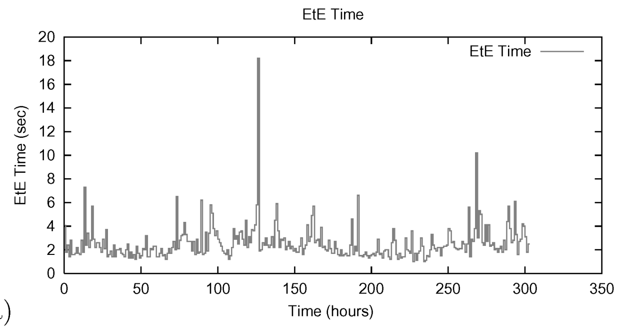
Fig. 12. HPL site: (a) server file hit ratio for *itanium.html*; (b) server byte hit ratio for *itanium.html*.

higher (i.e., more page objects are cached by network and browser caches), the response time measured by EtE monitor is lower. Again, active probing techniques cannot measure (or account for) the page caching efficiency to reflect the "true" end-to-end response time observed by the actual clients.

We now switch to the analysis of the OV-Support site. We will only highlight some new observations specific to this site.

Figure 13(a) shows the average end-to-end response time as measured by EtE monitor when downloading the site main page. This site uses JavaServer Pages technology for dynamic generation of the content. Since dynamic pages are typically more "compute intensive," this is reflected in a higher server-side processing fraction in overall response time. Figure 13(b) shows the network-server time ratio in the overall response time. It is higher compared to the network-server ratio for static pages from the HPL site. One interesting detail is that the response time spike around the 127 hour mark has a corresponding spike in increased server processing time, indicating some server-side problems at this point. The combination of data provided by EtE monitor can help service providers to better understand site-related performance problems.

The OV-Support site pages are composed of a large number of embedded images. Two most popular site pages, which account for almost 50% of all the page accesses, consist of 32 objects. The caching efficiency for the site is very high: only 8–9 objects are typically retrieved from the server, while the other objects are served from network and browser caches. The site server is running HTTP 1.0 server. Thus typical clients used 7–9 connections to retrieve 8–9

(a)


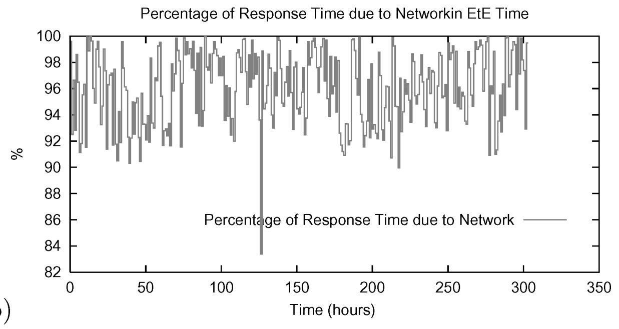
(b)


Fig. 13. OV-Support site during 2 weeks: (a) end-to-end response time for accesses to a main page; (b) percentage of response time due to network for the main page.

objects. The *ConcurrencyCoef* (see Section 6), which reflects the overlap portion of the latency between different connections for this page, was very low, around 1.038 (in fact, this is true for the site pages in general). This indicates that the efficiency of most of these connections is almost equal to sequential retrievals through a single persistent connection.

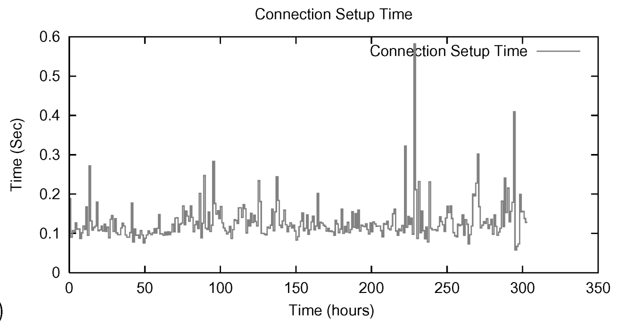
Figure 14(a) shows the connection setup time measured by EtE monitor. This connection setup time is a part of the latency for each of the 7–9 connections used to retrieve the corresponding Web page. Thus, the Web access for retrieval of k objects (let $k = 8$) at OV-Support site is represented by the following sequence of activities:

(1) a client sends a connection establishment request; once the connection is established, the client sends the request for the **first object**; the server sends a corresponding object to the client;

(2)–(7) sequential activities for establishing connections with the server for retrieving objects 2–7 (similar to the activity described in step 1);

(8) a client sends a connection establishment request; once the connection is established, the client sends the request for the **last embedded object**; the server sends a corresponding object to the client.

Clearly, the time for connection establishment at each object retrieval represents the unnecessary overhead, which can be avoided if the site server would run an HTTP 1.1 server, allowing clients to use persistent connections.



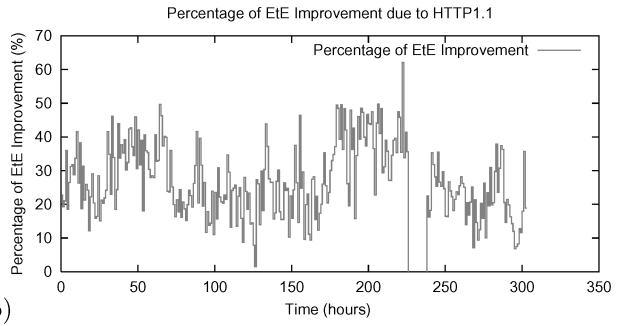


Fig. 14. OV-Support site during 2 weeks: (a) connection setup time for the main page; (b) an estimated percentage of end-to-end response time improvement if the server runs HTTP1.1.

We perform a simple computation: how much of the end-to-end response time observed by current clients can be improved if the site server would run an HTTP 1.1 server, allowing clients to use just two persistent connections to retrieve the corresponding objects from the site?

For example, if a client uses 8 different connections to retrieve 8 objects and the overlap portion of the latency between these different connections is very low (*ConcurrencyCoef*=1.038), then by replacing 8 different connections with 2 persistent connections, each retrieving 4 objects, the latency observed by the client can be improved by "6 connection setup delays."

Figure 14(b) shows the estimated percentage of end-to-end response time improvement available from running an HTTP 1.1 server. On average, during the observed interval, the response time improvement for *url1* is around 20% (2.6 sec is decreased to 2.1 sec), and for *url2* is around 32% (3.3 sec is decreased to 2.2 sec).

Figure 14(b) reveals an unexpected "gap" between 230–240 hour marks, when there was "no improvement" due to HTTP 1.1. More careful analysis shows that during this period, all the accesses retrieved only a basic HTML page using one connection, without consequent image retrievals. The other pages during the same interval have a similar pattern. It looks like the image directory was not accessible on the server. Thus, EtE monitor, by exposing the abnormal access patterns, can help service providers get additional insight into service related problems.

Table VI. EtE Monitor Performance Measurements

Duration, Size, and Execution Time	HPL site	OV-Support site
Duration of data collection	3 days	1 day
Collected data size	7.2 GB	8.94 GB
Transaction Log size	35 MB	9.6 MB
Entries in *Transaction Log*	616,663	157,200
Reconstructed page accesses	90,569	8,642
Reconstructed pages	5,821	845
EtE Execution Time	12 min 44 sec	17 min 41 sec

Finally, we present a few performance numbers to reflect the execution time of EtE monitor when processing data for the HPL and OV-Support sites. The tests are run on a 550Mhz HP C3600 workstation with 512 MB of RAM. Table VI presents the amount of data and the execution time for processing 10, 000, 000 TCP packets.

Reconstruction module performance depends on the complexity of the Web page composition. For example, the OV-Support site has a much higher percentage of embedded objects per page than the HPLabs pages. This "higher complexity" of the reconstruction process is reflected by the higher EtE monitor processing time for the OV-Support site (17 min 41 sec) compared to the processing time for the HPLabs site (12 min 44 sec). The amount of incoming and outgoing packets of a Web server farm that an EtE monitor can handle also depends on the rate at which *tcpdump* can capture packets and the traffic of the Web site.

7.2 IT-Support Case Study

The third site under study is a support site, which provides a variety of technical information and tools on software, hardware, and the network to help customers manage their multivendor computing environment. We call it *IT-Support* site. The architecture of this site is based on a Web server cluster with Cisco Distributed Director [Cisco Distributed Director www.cisco.com] for load balancing, using "sticky connections". We measured the site performance at one of the site's Web servers for 2 weeks, from March 25, 2002 to April 8, 2002.

The *IT-Support* site also uses Keynote [Keynote Systems, Inc. www.keynote.com] for performance monitoring. It is especially interesting for us to compare the Keynote measurements and the EtE monitor measurements to understand the differences in the two approaches for measuring Web service performance.

The Web pages published on this site are both dynamic and customized. The pages returned to the clients are dynamically generated based on a set of preferences and parameters customized to the end clients. For example, a client may select among 10 different language options for the site content. The page accessed via the same URL but with different language options might have a set of differently specialized, embedded objects and images. Thus, the URL of a particular page access has not only the information about the original URL of the accessed page but also the client parameters such as the session id,

Table VII. *At-a-Glance* Statistics for *IT-Support* Site During the Measured Period

Metrics	url1	url2	url3	url4
EtE time	3.2 sec	3.5 sec	4 sec	2.6 sec
% of accesses above 6 sec	7%	10.7%	14.3%	3.9%
% of aborted accesses above 6 sec	0.8%	1.9%	1.5%	1.1%
% of accesses from clients-proxies	5.3%	21.9%	6.8%	4.1%
EtE time from clients-proxies	6.4 sec	5.6 sec	7 sec	6.1 sec
% in EtE time due to network	86.1%	69.8%	47.6%	46.4%
Page access size	85.5 KB	62 KB	48.5 KB	58.3 KB
Number of requests	10.3	2	3.5	1.8
Number of connections	8.4	1.7	3.4	1.3

language option, and so on, as shown in the example below:

$\texttt{/x/y/z/doc.pl/sid} = \texttt{09e5998613f5dba43a} \mid \texttt{LANGUAGE}_{\texttt{O}}\texttt{PTION} = \texttt{japanese}$

So, each access to a logically identical URL is defined by a different URL expression. The service providers of this site provided us with a set of policies (regular expressions) on how to generate customized URLs, which are used to aggregate client's accesses to these URLs and measure the performance.

Table VII summarizes the site performance *at-a-glance* during the measured period using the four most frequently accessed pages, which represent four different support services at this site.

The average end-to-end response time of client accesses to these pages reflects good overall performance. However, in the case of *url1, url2*, and *url3*, a significant percentage of accesses take more than 6 sec to complete (7%–14%). The percentage of accesses issued by clients-proxies varies from 4.1% for *url4* to 21.9% for *url2*. We notice that the average response times for accesses from clients-proxies are commonly higher than the average response times observed by all the clients accessing those URLs.

The other distinctive feature for this workload is that the server processing portion in the overall response time is increased significantly compared to the previous two case studies, and it becomes a dominant component for *url3* and *url4*. For Web sites with dynamic content and complex multi-tier architectures, the latency breakdown to network-related and server-related portions is necessary for effective service assessment and future optimizations.

As mentioned earlier, the content returned for a particular URL depends on a language option and a set of other client specific parameters. Thus, each "logical" URL is an aggregation of many customized pages. In this case, the "size" of a Web page and the "corresponding set of embedded objects" are not uniquely identified. EtE monitor identifies a **combined set** of embedded objects during the construction of the knowledge base of Web pages, and uses this information to correctly reconstruct page accesses. As a result, some metrics measured by EtE monitor become meaningless, such as the average page size, the number of embedded objects, the file and byte hit ratios. However, service providers can use their knowledge about specific Web pages of interest to approximate the corresponding file and byte hit ratios based on the information reported by EtE monitor, such as the average size of client page accesses and the average number of requests for objects in a page.

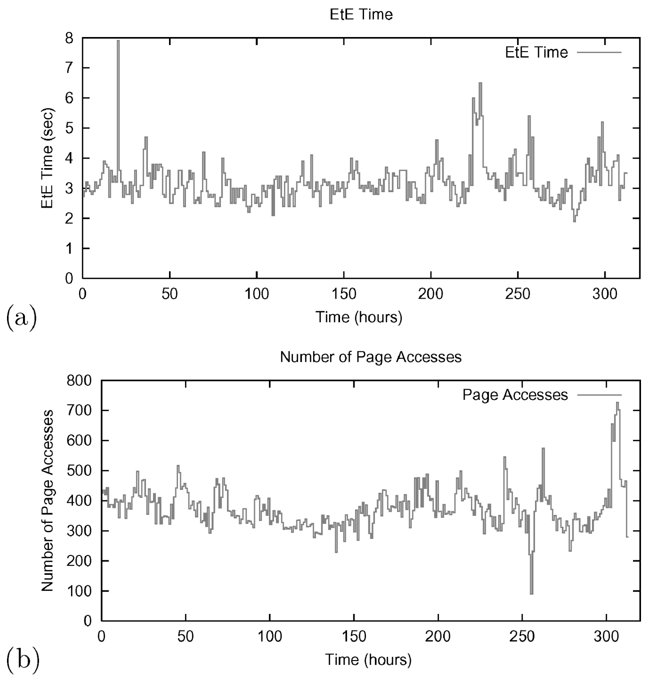


Fig. 15. *IT-Support* site: response time and number of page accesses to *url1* measured by EtE monitor.

Now, we compare the *IT-Support* site response time as measured by EtE monitor with Keynote service performance measurements.

Figure 15(a) shows the average response time (hourly) for page accesses to *url1* as measured by EtE monitor. Figure 15(b) reflects the number of page accesses to *url1*.

Figure 16 shows two Keynote graphs for the corresponding time period and the same server: the top graph represents the average response time (hourly) measured by Keynote agents probing *url1*; the bottom graph reflects the site availability (i.e. the percentage of probes successfully returned by the site).

The average response time reported by Keynote measurements is 4.6 sec for the corresponding time period. The *IT-Support* site deploys an additional, specially designed redirection mechanism for load balancing of the site traffic. The Keynote measurements of the redirection time was 1.1 sec on average (the DNS access time is also included in this measurement).[5] Thus, the average

[5]The current version of EtE monitor provides the measurements for successful responses with 200 status. EtE monitor functionality can be extended to measure the redirection time, i.e. the responses with 302 status. However, for the transactions with redirection, the redirection time constitutes only a portion of overall response time. A typical transaction with redirection consists of two parts: 1) the original client request with the corresponding redirection response by the site server, and 2) the second client request issued to a designated (by redirection) server. In order to measure the overall response time for such client requests one needs to correlate both of these transactions, which is a challenging problem even when the redirection is done within a single site.

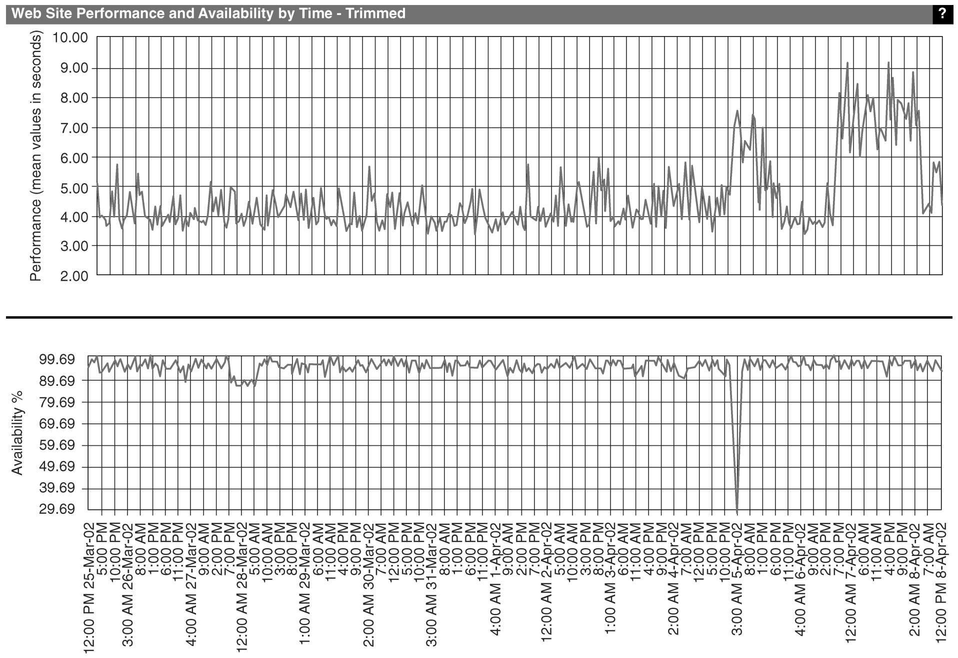


Fig. 16. *IT-Support* site: response time and site availability as measured by Keynote service for *url1*.

response time of the accesses to *url1* without redirection is 3.5 sec as measured by Keynote.

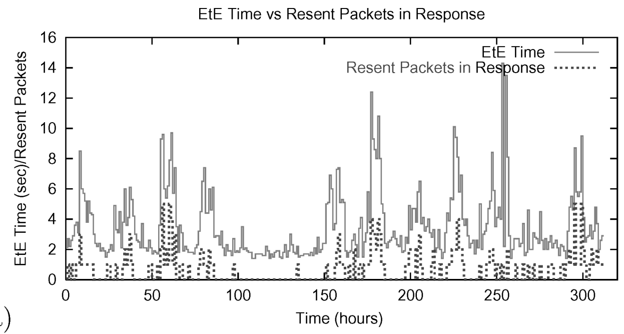
The EtE monitor was deployed on a single node of this cluster. The average end-to-end response time of client accesses to *url1* is 3.2 sec as reported by EtE monitor. Overall, the results of measurements provided by Keynote and EtE monitor match well. The slightly higher response time measured by Keynote may be due to downloading the whole page (with all the embedded images) from the original server all the time, while the real clients might benefit from network and browser caches.

The lower Keynote graph in Figure 16 reflects *IT-Support* site availability. It has a significant “dip” on April 5, at 3 am. The EtE monitor reflects a similar “dip” in the number of successful accesses to the site along the corresponding mark 256 in Figure 15(b).

Figure 17(a) shows the average response time for accesses to *url2* together with the number of resent packets during the corresponding hours. Most of the time, when the number of resent packets increases, the corresponding EtE time also increases. Figure 17(b) shows the network/server time ratio along with the number of resent packets (we scale the number of resent packets by 10 times to stress the similarity of patterns): the networking portion of response time tends to increase during the intervals when the number of resent packet increases.

Client population analysis is another area attracting service provider’s special interest. Knowledge about the largest client clusters and their response

(a)



(b)

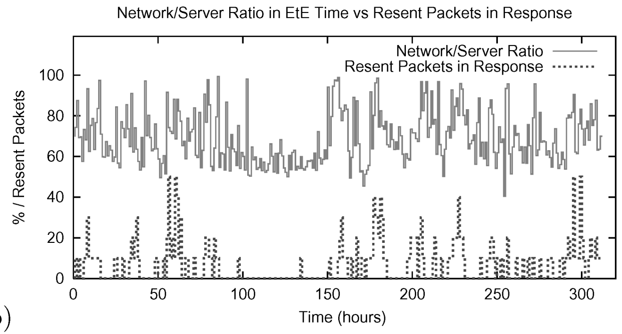


Fig. 17. *IT-Support* site: (a) average response time of accesses to *url2* and number of resent packets, (b) network/server time ratio in response time for *url2* and number of resent packets (multiplied by 10).

Table VIII. Percentage of the Client Accesses from the Asia-Pacific Region and End-to-End Response Times for these Accesses

Metrics	url1	url2	url3	url4
EtE time (All Clients)	3.2 sec	3.5 sec	4 sec	2.6 sec
EtE Time (Asia-Pacific Clients)	3.9 sec	4.7 sec	4.4 sec	6.7 sec
% of Asia-Pacific Clients Accesses	4.8%	6.1%	9.6%	0.3%

times is extremely useful for service providers to make wise decisions on additional server and cache placement [Krishnamurthy and Wang 2000].

EtE monitor can provide information about client clustering by associating them with corresponding ASes (Autonomous Systems). The service providers of *IT-Support* site have a special concern about their clients from the Asia-Pacific region (AS numbers between 9216-10239 represent Asia-Pacific domains). Table VIII shows the average response times for Asia-Pacific clients (AP clients) and percentage of their accesses to the four most popular *URL*s under study.

As, Table VIII shows, the AP client accesses constitute from 0.3% to 9.6% of all clients accesses for the four most popular *URL*s. The end-to-end response times observed by Asia-Pacific clients for *url1—url3* are only slightly higher than the corresponding average response times observed by all the clients, which was counter-intuitive.

(a)

(b)

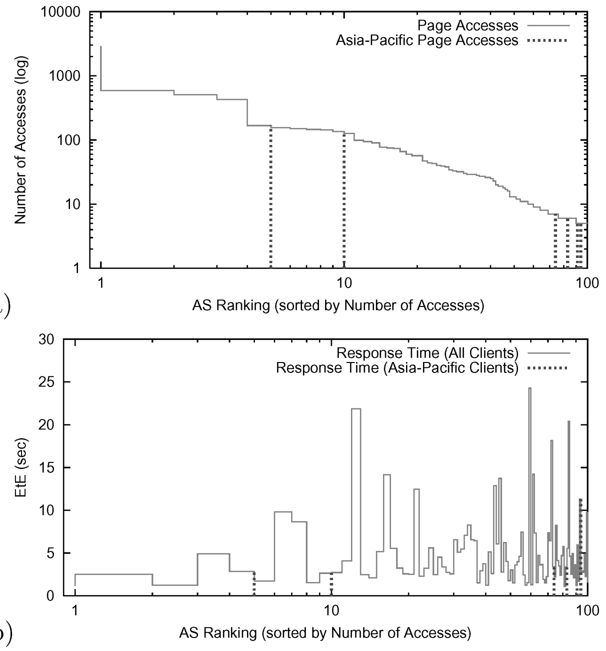


Fig. 18. *IT-Support* site, *url1*: daily analysis of 100 ASes with the largest client clusters: (a) number of different clients accessing the main page; (b) corresponding end-to-end response time per AS.

The following two figures, Figure 18 and Figure 19, present a daily snapshot of client population of those who accessed *url1* and *url2* on April 2, 2002 (Tuesday).

Figure 18 (Figure 19) shows the ASes sorted by the number of clients accesses from them. Note that both the X and Y axes use a log scale. The largest domain is responsible for 36.7% (12.6%) of accesses to *url1* (*url2*) during this day. We only show the first 100 domains, which represent 93.7% (95.5%) of all clients, accesses to this page. The long tail of remaining domains (from 100 to 300 domains) has only a few client accesses. In these figures, the dashed lines show the accesses from Asia-Pacific domains. Most of the response times for Asia-Pacific clients are not much different from the other US based clients.

This information provides a useful quantitative view of response times to the major client clusters. It can be used for site efficiency design to determine if a geographically distributed Web cluster is needed to improve site performance. Such information can also be used for content delivery networks to make appropriate decisions on data placement for a given client population.

The ability of EtE monitor to reflect a site performance for different ASes (and groups of IP addresses) happens to be a very attractive feature for service providers. When service providers have special SLA-contracts with certain groups of customers, EtE monitor provides a unique ability to measure the response time observed by those clients and to validate QoS targets for those contracts.

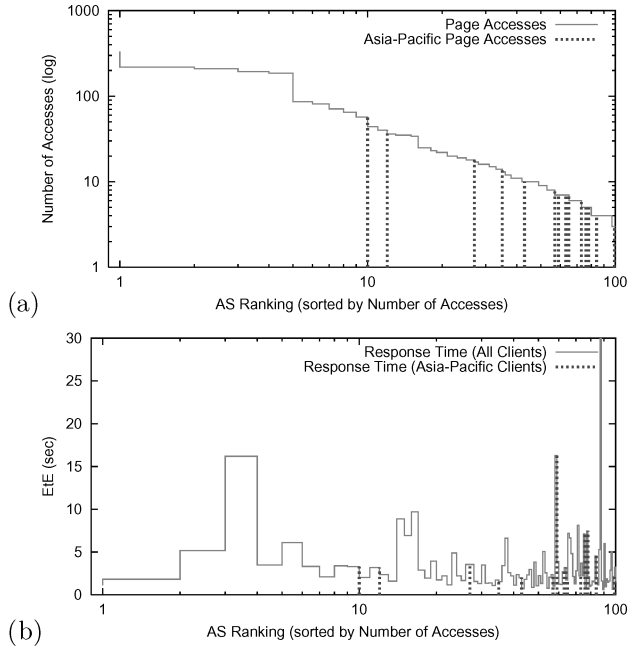


Fig. 19. *IT-Support* site, *url2*: daily analysis of 100 ASes with the largest client clusters: (a) number of different clients accessing the main page; (b) corresponding end-to-end response time per AS.

7.3 Validation Experiments

We performed two groups of experiments to validate the accuracy of EtE monitor performance measurements and its page access reconstruction power.

In the first experiment, we used two remote clients residing at Duke University and Michigan State University to issue a sequence of 40 requests to retrieve a designated Web page from the HPLabs external Web site, which consists of an HTML file and 7 embedded images. The total page size is 175 Kbytes. To issue these requests, we use *httperf* [Mosberger and Jin 1998], a tool that measures the connection setup time and the end-to-end time observed by the client for a full page download. At the same time, an EtE monitor measures the performance of the HPLabs external Web site. From EtE monitor measurements, we filter the statistics about the designated client accesses. Additionally, in EtE monitor, we compute the end-to-end time using two slightly different approaches from those discussed in Section 6.1:

—EtE time (*last byte*): where the *end* of a transaction is the time when the last byte of the response is sent by a server;
—EtE time (*ACK*): where the *end* of a transaction is the time when the ACK for the last byte of the response is received.

Table IX summarizes the results of this experiment (the measurements are given in *sec*):

The connection setup time reported by EtE monitor is slightly higher (14–15 ms) than the actual setup time measured by *httperf*, since it includes the

Table IX. Experimental Results Validating the Accuracy of EtE Monitor Performance Measurements

	httperf		EtE monitor		
Client	Conn Setup	Resp. time	Conn Setup	EtE time (last byte)	EtE time (ACK)
Michigan	0.074	1.027	0.088	0.953	1.026
Duke	0.102	1.38	0.117	1.28	1.38

Table X. Experimental Results Validating the Accuracy of EtE Monitor Reconstruction Process for HPL and OV-Support Sites

Metrics	HPL url1	HPL url2	OV-Support url1	OV-Support url2	IT-Support url1	IT-Support url2
Reconstructed page accesses (*filtered logs*)	36,402	17,562	17,601	11,310	109,553	31,844
EtE time (*filtered logs*)	3.3 sec	4.1 sec	2.4 sec	3.3 sec	3 sec	2.8 sec
Reconstructed page accesses (*masked logs*)	33,735	14,727	15,401	8,890	108,620	28,592
EtE time (*masked logs*)	3.2 sec	4.1 sec	2.3 sec	3.6 sec	2.9 sec	2.6 sec

time to establish a TCP connection as well as to receive the first byte of a request. The EtE time (*ACK*) coincides with the actual measured response time observed by the client. The EtE time (*last byte*) is slightly lower than the actual response time by exactly a round trip delay (the connection setup time measured by *httperf* represents the round trip time for each client, accounting for 74–102 ms). These measurements correctly reflect our expectations for EtE monitor accuracy (see Section 6.1). Thus, we have some confidence that EtE monitor accurately approximates the actual response time observed by the client.

The second experiment was performed to evaluate the reconstruction power of EtE monitor. The EtE monitor with its two-pass heuristic method actively uses the *referer* field to reconstruct the page composition and to build a *Knowledge Base* about the Web pages and objects composing them. This information is used during the second pass to more accurately group the requests into page accesses. The question to answer is: how dependent are the reconstruction results on the existence of *referer* field information. If the *referer* field is not set in most of the requests, how is the EtE monitor reconstruction process affected? How is the reconstruction process affected by accesses generated by proxies?

To answer these questions, we performed the following experiment. To reduce the inaccuracy introduced by proxies, we first filtered the requests with *via* fields, which are issued by proxies, from the original *Transaction Log*s for all of the sites under study. These requests constitute 24% of total requests for the *HPL* site, 1.1% of total requests for the *OV-Support* site, and 7% for the *IT-Support* site. We call these logs *filtered logs*. Further, we mask the *referer* fields of all transactions in the *filtered logs* to study the correctness of reconstruction. We call these modified logs *masked logs*, which do not contain any *referer* fields. We notice that the requests with *referer* fields constitute 56% of the total requests for the *HPL* site, 69% for the *OV-Support* site, and 35% for the *IT-Support* site in the *filtered logs*. Then, EtE monitor processes the *filtered logs* and *masked logs*. Table X summarizes the results of this experiment.

The results of *masked logs* in Table X show that EtE monitor does a good job of page access reconstruction even when the requests do not have any *referer* fields. However, with the knowledge introduced by the *referer fields* in the *filtered logs*, the number of reconstructed page accesses increases by 8–21% for the considered URLs in Table X. Additionally, we also find that the number of reconstructed accesses increases by 5.3–19.8% for all the considered URLs if EtE monitor processes the original logs without filtering either the *via* fields or the *referer* fields. The difference of EtE time between the two kinds of logs in Table X can be explained by the difference of the number of reconstructed accesses. Intuitively, more reconstructed page accesses lead to higher accuracy of estimation. This observation also challenges the accuracy of active probing techniques considering their relatively small sampling sets.

8. LIMITATIONS

There are a number of limitations to our EtE monitor architecture. Since EtE monitor extracts HTTP transactions by reconstructing TCP connections from captured network packets, it is unable to obtain HTTP information from encrypted connections. Thus, EtE monitor is not appropriate for sites that encrypt much of their data (e.g., via SSL). In the case where the encryption is not directly conducted by the Web server (e.g., by specialized hardware running on a dedicated node in "front" of Web server), the EtE monitor can be deployed either between the server and the encrypting hardware or directly at the server to overcome this problem.

In principle, EtE monitor must capture all traffic entering and exiting a particular site. Thus, our software must typically run on a single Web server or a Web server cluster with a single entry/exit point where EtE monitor can capture all traffic for this site. If the site "outsources" most of its popular content to CDN-based solutions then EtE monitor can only provide the measurement information about the "rest" of the content, which is delivered from the original site. For sites using CDN-based solutions, the active probing or page instrumentation techniques are more appropriate solutions to measure the site performance. A similar limitation applies to pages with "mixed" content: if a portion of a page (e.g., an embedded image) is served from a remote site, then EtE monitor cannot identify this portion of the page and cannot provide corresponding measurements. In this case, EtE monitor consistently identifies the portion of the page that is stored at the local site, and provides the corresponding measurements and statistics. In many cases, such information is still useful for understanding the performance characteristics of the local site.

The EtE monitor does not capture DNS lookup times. Only active probing techniques are capable of measuring this portion of the response times. While DNS resolution time may not be negligible in some cases, it is only experienced when a client first contacts the server and the resolution result is not cached locally. Subsequent page accesses do not experience any additional delay due to DNS.

Further, for clients behind proxies, EtE monitor can only measure the response times to the proxies instead of to the actual clients.

As discussed in Section 3, the heuristic we use to reconstruct page content may determine incorrect page composition. Although the statistics of access patterns can filter invalid accesses, it works best when the sample size is large enough.

Dynamically generated Web pages introduce another issue with our statistical methods. In some cases, there is no consistent *content template* for a dynamic Web page if each access consists of different embedded objects (for example, some pages use a rotated set of images or are personalized for client profiles). In this case, there is a danger that metrics such as the *server file hit ratio* and the *server byte hit ratio* introduced in Section 6 may be inaccurate. However, the end-to-end time will be computed correctly for such accesses.

There is an additional problem (typical for server access log analysis of e-commerce sites) about how to aggregate and report the measurement results for dynamic sites where most page accesses are determined by *URL*s with client customized parameters. For example, an e-commerce site could add some client specific parameters to the end of a common URL path. Thus, each access to the logically same Web page has a different URL expression. However, service providers may be able to provide the policy of how these URLs are generated. With the help of the policy description, EtE monitor is still able to aggregate these URLs and measure server performance.

9. CONCLUSIONS AND FUTURE WORK

Today, Internet services are delivering a large array of business, government, and personal services. Understanding the performance characteristics of Internet services is critical to evolving and engineering Internet services to match changing demand levels, client populations, and global network characteristics. From a client's perspective, Web site responsiveness corresponds to Web page download times. Typically, a Web page is composed of multiple objects: a main HTML file and several embedded objects such as images. However, HTTP does not provide any means to delimit the beginning or the end of a Web page to effectively measure the overall response time for Web page retrieval.

Existing tools for evaluating Web service performance typically rely on active probing to a fixed set of URLs or on Web page instrumentation that monitors download performance to a client and transmits a summary back to a server. The first approach does not provide a representative sample of current client access patterns while the second approach cannot breakdown the various network and server components associated with Web page retrieval. This paper presents EtE monitor, a novel approach to measuring Web site performance. Our system passively collects packet traces from the server site to determine service performance characteristics. We introduce a two-pass heuristic method and a statistical filtering mechanism to accurately reconstruct composition of individual page and performance characteristics integrated across all client accesses.

Relative to existing approaches, EtE monitor offers the following benefits: i) a breakdown between the network and server overhead of retrieving a Web page, ii) longitudinal information for all client accesses, not just the subset probed by

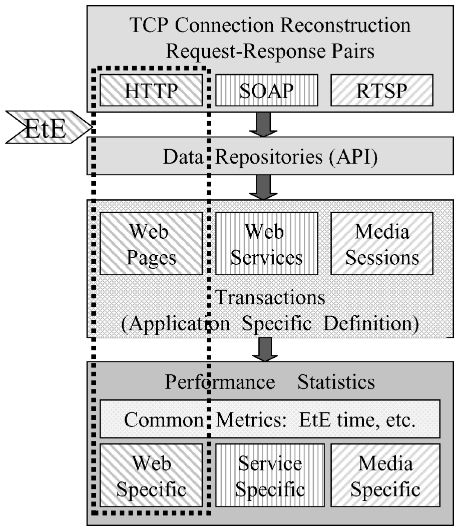


Fig. 20. General framework for passive monitoring and measurements.

a third party, iii) characteristics of accesses that are aborted by clients, iv) an understanding of the performance breakdown of accesses to dynamic, multi-tiered services, and v) quantification of the benefits of network and browser caches on server performance. Our initial implementation and performance analysis across three sample sites confirm the utility of our approach. The three case studies performed in this article cover the most typical Web site arrangements: from sites comprised of static content to the sites represented by highly dynamic and personalized pages. One important contribution of our work is a demonstration of our tools generality to dynamically-generated content. For such services, active probing techniques cannot determine a representative workload while techniques using Web-page instrumentation cannot break down the location of access bottlenecks, for example, between the server CPU (an increasingly important component for dynamic content) and the wide-area network.

We are currently investigating the use of our tool to understand client performance on a per-network region. This analysis can aid in the placement of wide-area replicas or in the choice of an appropriate content distribution network. Finally, our architecture is general to analyzing the performance of multi-tiered Web services. For example, application-specific log processing can be used to reconstruct the breakdown of latency across multiple tiers for communication between a load balancing switch and a front-end Web server, or communication between a Web server and the storage or database tier. One of the challenging directions for future research is in using the performance data provided by EtE monitor for intelligent decision making for active management of Web sites. In recent work [Krishnamurthy and Wills 2002; Krishnamurthy et al. 2002], authors classify Web site clients by using the response time these clients experience. Then authors the demonstrate how the server can alter the delivered content/policy/caching decisions in order to improve the response time observed by clients.

The other opportunity we plan to investigate is whether our approach can be extended to design a more general framework for passive monitoring and measurements of a broader set of network services such as Web services and media services as shown in Figure 20.

Most of the network services use application-specific protocols, for example, SOAP or RTSP, and use TCP/IP for their transport services. Some of the basic modules designed in EtE monitor, such as the TCP connection reconstruction module, can be reused in the protocol/application specific extraction modules. The notion of transaction might be defined accordingly in an application-specific reconstruction module (for example, in EtE monitor, the unit of such a transaction is a Web page while it might be a series of RPCs in another service). Correspondingly, the performance analysis module would extract the service-specific metrics of interest. We believe such a "plug-and-play" modular framework might lead to a monitoring and measurement utility service general to a broad range of network services.

ACKNOWLEDGMENTS

Both the tool and the study would not have been possible without the generous help of our HP colleagues: Mike Rodriquez, Steve Yonkaitis, Guy Mathews, Annabelle Eseo, Peter Haddad, Bob Husted, Norm Follett, Don Reab, and Vincent Rabiller. Their help is highly appreciated. Our special thanks to Claude Villermain who helped to identify and to correct a subtle bug for dynamic page reconstruction. Our sincere thanks to Jason Nieh and Craig Wills for useful remarks and numerous suggestions during several thorough revisions of the paper. We would like to thank the anonymous referees for insightful questions and constructive criticism, which helped to significantly improve the content and presentation of the paper.

REFERENCES

Barford, P. and Crovella, M. 2000. Critical Path Analysis of TCP Transactions. In *Proceedings of SIGCOMM*.

Caceres, R., Duffield, N., Feldmann, A., Friedmann, J., Greenberg, A., Greer, R., Johnson, T., Kalmanek, C., Krishnamurthy, B., Lavelle, D., Mishra, P., Ramakrishnan, K., Rexford, J., True, F., and van der Merwe, J. 2000. Measurement and Analysis of IP Network Usage and Behaviour.

Candle Corporation: eBusiness Assurance. http://www.candle.com/.

Cardwell, N., Savage, S., and Anderson, T. 2000. Modeling TCP Latency. In *INFOCOM*.

Cisco Distributed Director. http://www.cisco.com/.

Feldmann, A. 2000. BLT: Bi-Layer Tracing of HTTP and TCP/IP. In *Proceedings of WWW-9*.

Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and Berners-Lee, T. 2001. Hypertext Transfer Protocol—HTTP/1.1. Tech. Rep. RFC 2616, IETF. June.

Gomez, Inc. http://www.gomez.com.

Hellerstein, J. L., Maccabee, M. M., Iii, W. N. M., and Turek, J. J. 1999. ETE: A Customizable Approach to Measuring End-to-End Response Times and Their Components in Distributed Systems. In *International Conference on Distributed Computing Systems*.

HP Corporation. OpenView Products: Web Transaction Observer. http://www.openview.hp.com.

IBM Corporation. Tivoli Web Management Solutions. http://www.tivoli.com/products/demos/twsm.html.

IBM Research. Page Detailer. http://www.research.ibm.com/pagedetailer/.

JavaServer Pages. http://java.sun.com/products/jsp/technical.html.

JavaServlet Technology. `http://java.sun.com/products/servlet/`.

Keeley, T. 2000. Thin, High Performance Computing over the Internet. Eight International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2000).

Keynote Systems, Inc. `http://www.keynote.com`.

Krishnamurthy, B. and Rexford, J. 1998. Software Issues in Characterizing Web Server Logs.

Krishnamurthy, B. and Rexford, J. 1999. En Passant: Predicting HTTP/1.1 Traffic.

Krishnamurthy, B. and Rexford, J. 2001. *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement*. Addison Wesley.

Krishnamurthy, B. and Wang, J. 2000. On Network-Aware Clustering of Web Clients. In *Proceedings of ACM SIGCOMM 2000*.

Krishnamurthy, B., Wills, C. E., , Zhang, Y., and Vishwanath, K. 2002. Design, Implementation, and Evaluation of a Client Characterization Driven Web Server. In *Proceedings of WWW*.

Krishnamurthy, B. and Wills, C. E. 2000. Analyzing Factors That Influence End-to-end Web Performance. In *Proceedings of WWW*.

Krishnamurthy, B. and Wills, C. E. 2002. Improving Web performance by client characterization driven server adaptation. In *Proceedings of WWW*.

Mosberger, D. and Jin, T. 1998. Httperf—A Tool for Measuring Web Server Performance. *Performance Evaluation Review 26*, 3 (December).

NetMechanic, Inc. `http://www.netmechanics.com`.

NetQoS Inc. `http://www.netqos.com`.

Nielsen, H. F., Gettys, J., Baird-Smith, A., Prud'hommeaux, E., Lie, H. W., and Lilley, C. 1997. Network Performance Effects of HTTP/1.1, CSS1, and PNG. In *Proceedings of SIGCOMM*.

Olshefski, D. P., Nieh, J., and Agrawal, D. 2001. Inferring Client Response Time at the Web Server. In *Proceedings of USITS*.

Porivo Technologies, Inc. `http://www.porivo.com`.

Rajamony, R. and Elnozahy, M. 2001. Measuring Client-Perceived Response Times on the WWW. In *Proceedings of the Third USENIX Symposium on Internet Technologies and Systems (USITS)*.

Seshan, S., Stemm, M., and Katz, R. H. 1997. SPAND: Shared Passive Network Performance Discovery. In *Proceedings of USITS*.

Smith, F. D., Campos, F. H., Jeffay, K., and Ott, D. 2001. What TCP/IP Protocol Headers Can Tell Us About the Web. In *Proceedings of ACM SIGMETRICS*.

Software Research Inc. `http://www.soft.com`.

Stemm, M., Katz, R. H., and Seshan, S. 2000. A Network Measurement Architecture for Adaptive Applications. In *Proceedings of IEEE INFOCOM*.

Tcpdump. `http://www.tcpdump.org`.

Received July 2002; revised January 2003; accepted March 2003