Improving Web Performance by Client Characterization Driven Server Adaptation

Balachander Krishnamurthy
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ
bala@research.att.com

Craig E. Wills
WPI
100 Institute Road
Worcester, MA
cew@cs.wpi.edu

ABSTRACT

We categorize the set of clients communicating with a server on the Web based on information that can be determined by the server. The Web server uses the information to direct tailored actions. Users with poor connectivity may choose not to stay at a Web site if it takes a long time to receive a page, even if the Web server at the site is not the bottleneck. Retaining such clients may be of interest to a Web site. Better connected clients can receive enhanced representations of Web pages, such as with higher quality images.

We explore a variety of considerations that could be used by a Web server in characterizing a client. Once a client is characterized as poor or rich, the server can deliver altered content, alter how content is delivered, alter policy and caching decisions, or decide when to redirect the client to a mirror site. We also use network-aware client clustering techniques to provide a coarser level of client categorization and use it to categorize subsequent clients from that cluster for which a client-specific categorization is not available.

Our results for client characterization and applicable server actions are derived from real, recent, and diverse set of Web server logs. Our experiments demonstrate that a relatively simple characterization policy can classify poor clients such that these clients subsequently make the majority of badly performing requests to a Web server. This policy is also stable in terms of clients staying in the same class for a large portion of the analysis period. Client clustering can significantly help in initially classifying clients for which no previous information about the client is known. We also show that different server actions can be applied to a significant number of request sequences with poor performance.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Internet*; H.5.3 [**Information Systems**]: Information Interfaces and Presentation—*Web-based interaction*

General Terms

Measurement, Performance

Keywords

client connectivity, client characterization, server adaptation

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

1. INTRODUCTION

Web performance has been a key focus of research over the last few years. User-perceived latency has a strong bearing on how long users would stay at a Web site and the frequency with which they return to the site. A Web site that is trying to retain users thus has a strong incentive to reduce the "time to glass" (the delay between the browser click and the delivery and display of the resource on the user's screen). For Web sites that have a critical need to retain users beyond the first page there is a strong motivation to deliver the content quickly to the user. Given the vagaries of network delays, presence of intermediaries, and the user's network connectivity, the server has a strong incentive to deliver the most suitable (dynamically generated or statically selected) content quickly to the user.

Our work focuses on *learning* about the quality of the connection between the client and server to aid the server in making an informed decision on what content to serve. We seek to obtain the client's connectivity information based on information already available at the server. An alternative to this passive approach is to actively gather information about the more dynamic components of the end-to-end delay, such as the bandwidth of the client or delays in the network. This would however require considerable amount of active information gathering in different locales.

A motivation for our work is to concentrate on performance issues that are not due to the server itself. Server-induced delays can be reduced by improved scheduling policies or simply upgrading the server. We focus on other reasons behind a client experiencing poor performance such as low bandwidth, high latency, network congestion, delay at intermediaries between the client and server, and a slow client. A server may not be able to isolate the reasons for a given client experiencing poor performance but it can take remedial action in selecting a lower quality version of the resource or by serving the content in a different manner.

A Web site may have multiple variants of the same resource. If the Web server on the site *knew* that the client had poor (or rich) connectivity, then a more appropriate variant of the resource could be sent in the response. Alternately, the server could deliver the same content in a different way. Such a response might result in the user being more satisfied with the delivery speed or the quality of the response and help retain the client for the Web site.

The Web site does not need to know the precise level of connectivity of the client—it is enough to be able to classify the client into one of a few classes such as poor, normal, and rich. The number of alternate versions of the response that can be sent back is also not likely to be more than a few. By mapping the versions of the resource in advance to the different categories of a clients, the appropriate response can be sent shortly after client categorization.

For example, the server can choose between sending only the base document, the base document plus a few embedded resources, or sending the full container document. Apart from simply sending a different response, by identifying the client, the server can use the information to guide a variety of its policies. For example, the server might decide to keep a HTTP persistent connection open longer with clients who have poor connectivity to reduce their need to set up a new TCP connection. Or, the server can piggyback cache related information to reduce the need for future validation requests. In this work we explore the potential applicability of each of these actions.

Thus the challenge is to be able to classify clients in a stable manner into a few meaningful categories and enumerate a set of ways in which the classification can be used for meaningful improvements. Since a given Web site's content or access pattern may or may not lend itself to benefiting from such a classification, we first need to measure the proportion of retrievals that can benefit. Similarly, the number of clients who can benefit is another metric to gather. Once a client has been characterized, we need to map it to an appropriate action for the server to take. Well-connected clients may require no change in the actions taken by a server. Additionally, we can examine grouping clients for classification and application of server actions. For example, grouping via network-aware clustering [14] is a technique we explore.

The contributions of this paper are as follows:

1. We propose taking advantage of information already available to a Web server to classify client connectivity.

2. Based on the stability of the client classification and the degree to which it can be trusted, we propose a range of server actions that can be taken.

3. We evaluate the feasibility of our proposed client classification policies and server actions by examining a heterogeneous collection of Web server logs.

The rest of the paper is organized as follows: Section 2 discusses the various ways in which we can categorize clients and the range of available information on which to base this decision. Section 3 enumerates the range of server actions possible for the different categories of clients. Section 4 presents our methodology for categorizing clients and evaluating the potential applicability of the various server actions. Section 5 presents the results of our experiments carried out using a heterogeneous collection of large server logs. We conclude with a discussion of related work, a summary and plans for future work.

2. CLIENT CHARACTERIZATION

The first step in being able to adapt content or its manner of delivery for a client is to identify the client's characteristics. One approach is for clients to identify their characteristics themselves. Clients do this to a limited extent already by specifying the content types they are willing to accept. In addition, clients could specify their network connectivity such as dial-up, cable modem, T-1, etc., as they do now when they begin using a multimedia player. The clients can indicate their connectivity information via the CC/PP (Composite Capabilities/Preferences Profiles [8]) mechanism. CC/PP allows user agents and proxies to specify their capabilities and enables HTTP content negotiation (Section 7.9 of [13]) with servers. The problem with these approaches is that even if available, many clients may not use them for normal Web browsing. In addition, a client's experience may vary depending on the time of day or the number of network hops between the client and the server.

In the absence of explicit classification information from the client, it is useful for a Web server to be able to characterize a client. There are a number of potential pieces of information available to a server for such classification. We consider three types of classification based on network connectivity, response time and other considerations.

2.1 Network Connectivity

The first type of information involves characterizing the nature of the network connectivity between a client and a server. Ideally the server would like to know the round-trip time (RTT), bandwidth, and the amount of congestion in the path to the client. In practice, the Web server can only make inferences based on the network traffic it receives from the client. For example, the server can estimate the RTT by measuring the time between accepting a TCP connection from a client and receiving the first byte of the subsequent HTTP request. This interval requires a single IP packet round trip. This approach is simple and introduces no additional network traffic, although typical Web servers would need to be instrumented to measure this value.

While each TCP connection made by a client to the server can be used to refine the RTT estimation for the client by the server, it is more difficult to estimate bandwidth characteristics for the network connection. Tools that estimate bandwidth between two nodes typically work by sending packets of different sizes between the nodes and measuring the respective round trip times. Examples include *bing* [5], *pathchar* [11] and measuring end-to-end bulk transfer capacity [2]. This approach would add overhead for both the Web server and the client.

Another approach that the Web server can use to estimate the RTT at the HTTP-level is to initially respond to a client request with a HTTP redirect response (`302 Found`), which would typically cause the client to retrieve the redirected content. The difference between the two request times can then be used to estimate the RTT between successive HTTP requests. Unfortunately this approach introduces an additional HTTP transaction and further lengthens the response time for the client.

A different approach at the HTTP level uses the fact that browsers typically automatically request the set of embedded objects for the page after loading the container object for a Web page. Measuring the time between the retrieval of the base object and the first requested embedded object can be used to estimate the time between successive HTTP requests without necessitating additional requests or redirection solely for the purpose of measuring.

2.2 Response Time

Each of the previous measures seeks to characterize the RTT or bandwidth of the network connection between client and server. An alternate approach for classifying a client is to focus on the response time seen by the client and not be directly concerned with the nature of the network connection. This outcome-oriented characterization focuses less on the specific causes of poor performance and more on identifying the situations where it occurs.

One approach for estimating response time for a Web page is to again use the fact that typically browsers automatically download embedded objects on a page. Using this observation, the measured time between serving the base object and the last embedded object on the Web page approximates the total response time of the page for the client. This measurement can be done without introducing any additional Web traffic.

A more accurate estimate of the total delay experienced by the client (vs. the server) can be obtained by instrumenting a page with Javascript, which executes within the client's browser, to measure the total download time and report it back to the Web server [22]. Unfortunately this approach requires explicit instrumentation of pages and introduces additional HTTP traffic combined with the requirement that browsers accept Javascript. The Web server can also measure the number of connection aborts and resets from the client which may suggest the presence of poorly-connected, impatient clients [7].

2.3 Additional Factors for Characterization

There are several other factors that could be used in classifying a client. Information in the client request header such as accepted content types, the HTTP protocol version [12], and the client software itself could all be exploited in classifying the capabilities of the client. Once a client has been classified, this classification could be stored in cookies generated by a server and included by the client in subsequent requests to that server.

Browser or proxy caching can also affect the classification of a client. For example, if a client requests only a few of the embedded objects on a Web page that is known to have tens of embedded objects, a reasonable inference is that the client or an intervening proxy has many of the objects cached. Clients with or behind an effective cache may be considered richer if many needed objects are already in the cache. On the other hand, a proxy in the path may introduce additional delay for requests from a client to the Web server and make that client appear poorer.

A final consideration is identifying *which* clients to classify. The most obvious candidates for classification are those for whom response time is important—users employing browsers. In contrast, automated clients such as spiders should be filtered out and not be considered for classification. Recent work has examined the detection of spiders at a Web server [14, 3].

3. POTENTIAL SERVER ACTIONS

There a number of possible actions that a server could potentially take after characterizing a client. There are two broad classes of actions: those that alter the content, which can be applied for either rich or poor clients; and those that alter how the content is delivered, which are primarily applicable for poor clients. In deciding which action to take the server can use characteristics of the client or consider characteristics of a client group, such as those formed by network-aware clusters [14].

3.1 Alteration of Content

Given a range of content variants, a server could choose a larger and potentially enhanced variant to serve to richer clients and a reduced version for poorer clients. The server could provide a reduced version by including fewer, if any, embedded objects or by including "thinner" variants of embedded images.

3.2 Selection of Replica or Mirror

The first action a server performs when it receives a request is to decide if it is going to be handled on the machine where the request arrived. Often busy Web sites have a large collection of machines behind switches or redirectors where the content may be stored or generated. Popular search engines and other busy news sites use this approach. The front end server can use client's connectivity information to guide selection of the back-end machine if the content stored or generated in those machines is partitioned on this basis. After choosing the proper mirror, the manner in which the Request-URI is mapped to a specific resource can be additionally tailored based on the client's connectivity.

Additionally, sites that use Content Distribution Networks (CDNs), may have some resources delivered from mirror sites distributed on the Internet. Often the choice of a particular mirror site is left to the CDN; however, the origin server can provide hints about the client's connectivity which can be used by the CDN in selecting the mirror.

3.3 Alteration of Meta-Information

Well-connected clients and proxies are more likely to prefetch or prevalidate documents to lower user-perceived latency. The heuristic assignment of expiration meta-information to resources can be tailored by servers to ensure that poorly connected clients can avoid needless validation of less-frequently changing resources. Origin servers could issue hints via headers so that proxies between poorly connected clients and the origin server can increase the freshness interval for resources. By providing information to caches along the path, the origin server can help guide caching policies at the intermediaries.

As discussed in earlier work [15, 16, 10], hints can be provided by the server about request patterns of resources. Grouping resources into volumes and tailoring them to suit the client's interests can effectively provide useful information. Now, an additional factor can be used by the server in deciding the set of hints to be delivered to the client. A richly connected client might receive extended set of hints while a poorly connected client might receive a more suitably tailored set of hints or no hints at all.

3.4 Altering Manner of Delivery of Content

After a particular resource variant has been selected, the server can still control the manner in which it is delivered. The response can be compressed with a suitable compression algorithm that takes into account the connectivity information. Another way to reduce content is to send the difference between versions of resources for delta-enabled clients [20, 18]. Poorly connected clients can benefit even more from the delta mechanism since CPU speed and disk costs are improving faster than network latencies. A final alternative is for the server to bundle the embedded objects into a single resource, which can be retrieved by clients to avoid multiple rounds of requests to fetch the objects [25]. Yet another alternative is to combine this technique with compression to reduce the size of the bundle and use a delta-encoding mechanism to prevent bundling of objects already present in the client's cache.

3.5 Guiding Policy Decisions at the Server

Once a server has returned a response, its ability to use connectivity information does not end. In HTTP/1.1, connections between a client and a server can persist beyond a single request-response exchange. The HTTP protocol does not offer any guidelines on deciding when a persistent connection should be closed (see Section 7.5.5 of [13]). A server can decide when to close a persistent connection based on a variety of factors such as fairness, potential of future connections from the same client, duration for which connection was already open, etc. If a server knows that a client has poor connectivity, it might wish to keep the connection open longer than usual to ensure that the client does not pay the overhead of having to tear down and set up a new connection. A richly connected client could afford the overhead of setting up a new connection. Additionally, the server can assign a higher priority to poorly connected clients so that their requests are handled faster to reduce overall latency.

4. METHODOLOGY

Rather than instrument a Web server to characterize clients and evaluate the potential of taking different actions, the initial approach we have taken is to analyze the logs from a variety of Web server sites. By using logs for the analysis we can examine a wider variety of sites. We take advantage of the fact that client browsers are typically configured to automatically download embedded objects for a container page. We can thus use the delays between when the container page and subsequent embedded objects are served as a measure of the connectivity between a client and the server.

For our study, we assembled a heterogeneous collection of recent server logs ranging in duration and number of accesses. The data included logs from a university department, a specialized medical site, a large commercial company, a research organization, and a sporting site mirrored in multiple countries.

4.1 Log Analysis

Not all of the logs we used were in the same format, but all of the logs contained the following fields for each logged request: the requesting client (identified by IP address or string representation), the Request-URI, the response code, the date and time when the request was completed, and the response size. In addition, some of the logs also contained referrer and user-agent fields, which specify the URI that referred the current request and the agent software that made the request.

The first step in our analysis was to use the Request-URI to identify requests indicating an HTML object or the output of a server-side script. We marked these requests as container documents and counted the number of requests for each of them. We then determined the set of pages that account for 70% of accesses within the log to constitute the *popular* set of pages at that site. We focused further analysis only on these pages since the benefits for our approach are most likely to be applied to the most popular pages.

After preprocessing the log to identify the popular set of pages, we focused on using requests with `200 OK` or `304 Not Modified` HTTP response codes in the logs to identify *sequences* of requests. We define a sequence as a set of consecutive requests for a base object and at least one embedded object from the same client with the following characteristics:

- The first (base) request in the sequence returns HTML content based on Request-URI indicating an HTML object or the output of a server-side script.
- Each subsequent request in the sequence indicates the Request-URI is for an image, style sheet or Javascript object.
- If the referrer field is available in the log then the referrer field for the embedded objects must also match the URI of the base object. This additional check helps to eliminate a relatively small number of sequences where the embedded objects did not match with the base object.
- Finally, we set an arbitrary maximum threshold of 60 seconds between any object and the time the base object is downloaded. Any sequence that spans this duration is already going to be classified as poor and making the threshold larger increases the possibility of grouping requests for objects on different pages within the same sequence.

This analysis is not guaranteed to obtain all sequences of a base page followed by its set of embedded objects. For example, it will not work for pages with embedded HTML frames. It may also fail when multiple clients behind the same proxy are making requests at the same time to the server. However, it is not critical that we identify all sequences, but merely a sufficient number to characterize clients. The analysis produces sequences such as the sample shown in Table 1 where the retrieval of a base object of 12221 bytes is followed by the retrieval of 10 embedded objects over the course of 28 seconds. Two of the requests resulted in a response code of `304 Not Modified` for which no content bytes were returned. While a finer granularity than one second would be preferable, this coarse granularity does provide enough information to identify the relative duration of a sequence, which is the focus of our work.

Table 1: Sample Sequence

Object Number	Response Code	Content Size	Delay (in sec.) From Base Object
0	200	12221	-
1	200	183	2
2	200	1577	2
3	304	0	3
4	200	3322	4
5	200	2133	4
6	200	898	7
7	200	2803	8
8	304	0	11
9	200	400	11
10	200	2803	28

By focusing on clients who download the embedded objects on a page, we typically filter out spiders. However as an additional check, if the agent field is available we remove all spider requests that we can identify (e.g., Googlebot, Scooter etc.). However, subsequent analysis shows such filtering has little effect on the number of sequences we identify in a log.

As an additional analysis tool, we also clustered the set of unique IP addresses in each log using the technique outlined in [14]. This clustering was carried out by a small C program that uses a fast longest prefix matching library. This software is capable of clustering millions of IP addresses using a large collection of BGP prefixes (over 441K unique prefixes) in a few seconds. In cases where the logs recorded domain names versus IP addresses, we performed a DNS lookup on each name prior to clustering. In cases where the DNS lookup failed or the clustering technique was unable to cluster a client with other clients then that client was treated as its own cluster for subsequent analysis.

4.2 Client Characterization

Once we identify the set of sequences in a log, we use the characteristics of the sequences for a client to characterize that client. We choose three simple categories for characterizing a client: poor, normal and rich. We use the term "poor" to refer to clients who generally exhibit long delays between retrieval of the base object and subsequent embedded objects. In contrast, we use the term "rich" to refer to clients who consistently exhibit little delay between retrieval of the base object and subsequent embedded objects. We use the term "normal" to refer to clients who cannot be classified as either poor or rich.

We used two metrics in classifying a client: the delay between the base object and the first embedded object and the delay between the base object and the last embedded object in a sequence. The first metric measures the time for a client to receive at least part of the base object, parse it for embedded objects and for the server to receive and process the request for the first subsequent object. The second metric corresponds to how long a client must wait for all of the embedded objects for a page to be downloaded. While this value will vary according to the size and number of objects, it is a reasonable measure in identifying the delay characteristics of clients.

For each of these two metrics, we defined cumulative values E_{first} and E_{last} to represent long-term estimates for these two metrics for each client. To both minimize the amount of information stored for each client and to give greater weight to more recent history, we chose to use a exponentially weighted mean where the value for E_{first} (E_{last} is similarly defined) is given as:

$$E_{first} = \alpha E_{first} + (1 - \alpha) E_{measured}$$

where $E_{measured}$ is the current measurement for the delay between the base and first embedded object. The value α is the weighting factor and takes on values between zero and one. In our study we experiment with three values of α: 0.0, 0.3, and 0.7. Note that in the case where α=0.0 only the last measurement is used in classifying the client.

Using these defined values for E_{first} and E_{last} we define thresholds for what identifies a poor and rich client. Building on work from [17], Nielsen suggests download times of greater than 10 seconds causes discontinuities for a user [21]. Bickford suggests that users are no longer willing to wait beyond 8.5 seconds [6]. Chiu suggests more than 25 seconds is slow, but bases this threshold on a 56K connection speed [9]. Rather than define such fixed thresholds in our study, we use results of these studies as guidelines in identifying three sets of thresholds for the definition of a client experiencing poor performance (all times are in seconds):

1. if $E_{first} > 3$ or $E_{last} > 5$
2. if $E_{first} > 5$ or $E_{last} > 8$
3. if $E_{first} > 8$ or $E_{last} > 12$

We defined and explored only one threshold set for what defines a client experiencing good performance and hence warrants classification as a rich client:
if $E_{first} <= 1$ and $E_{last} <= 2$.

We used these thresholds to define whether a client was poor or rich (or normal if it met neither criteria). We also explored two levels of confidence with each policy. The first confidence level was more aggressive in classifying clients as rich or poor immediately after the first sequence was processed for a client. The second confidence level was more conservative in keeping the classification of a client as normal until at least six sequences were processed from a client.

The use of client clustering was also explored in our study where we use the accesses from all clients within a cluster to characterize a cluster as poor, rich or normal. When the initial request sequence is received from a client the default is to always classify that client as normal. However, in cases where an initial sequence is requested by a client who is part of a cluster seen earlier then the current classification for that cluster is assigned as the initial client classification.

We also recognize that a busy Web server cannot realistically store state about clients over a long period of time. We use garbage collection to remove the state and classification for clients when the last page access is more than one day old in the logs. We note, however, that it might make sense to retain information longer on a per-cluster basis.

Four parameters—the weighting factor, the thresholds for defining a poor client, the confidence level and the use of client clustering—were studied in this work. Table 2 summarizes the 36 combinations we studied.

Table 2: Summary of Policy Parameters

Parameter	Range of Values
Weighting factor α	0.0, 0.3, 0.7
Poor threshold (E_{first},E_{last})	(3,5), (5,8), (8,12)
Confidence level for client accesses	1, 6
Use of client clustering	no, yes

4.3 Evaluation of Characterization Policies

We used two approaches for evaluating the success of client characterization policies for classifying clients. First we examine the nature of sequences requested by poor and rich clients. In an ideal world, a server would like to know when the performance of serving a sequence is going to be bad and take mitigating action to improve the performance. A server would also like to know if the performance of serving a sequence is going to be good, which would potentially allow it to enhance the content it might serve to the client. Using these ideals, we define a client classification policy as successful if a high percentage of bad accesses in a log are done by clients classified as poor and a high percentage of good accesses in a log are done by clients classified as rich. The higher these percentages, the more possibilities for the server to take action, although the aggressiveness of the classification must also be tempered by potential negative effects of incorrect classification. For example, serving enhanced content to a client classified as rich, who is actually not rich may lead to negative consequences. In analyzing various classification policies, we will examine the tradeoffs between more and less aggressive classification policies.

Second, we keep track of the identity of clients who can benefit from our analysis. We estimate the percentage of clients who are *stable*; i.e., ones that either never or rarely move between the categories. The more stable the classification for a client, the more likely is it that a server can take tailored action. Clients with unstable classifications are not good candidates for server actions.

4.4 Applicability of Potential Server Actions

In Section 3 we identified a number of potential actions that a Web server could apply if it has characterized a client as poor or rich. In addition to using the Web server logs to classify clients, we also identified poorly performing sequences and determined which of these actions could be applied to these sequences.

A variety of considerations should be taken into account by a server before deciding on the applicability of a particular action to a sequence. Among the considerations are the length of the sequence (some sequences may have to be long before they are considered useful enough to trigger a particular server action), the size of the full container document, or the size of the base page. In Table 3, we list a series of possible server actions and the sequence criteria to be examined for each action.

The first action of reducing the amount of content can be applied to any page, but is most likely to improve performance for pages containing many images or bytes. However, this action may not be satisfactory because it changes the content of the page. Redirecting the client to a mirror via HTTP or DNS redirection can also be applied to any page, but it is most effective relative to other server actions when the bad performing sequence is due to a long RTT between client and server. We thus flag all sequences with a large delay for the first object.

Altering the meta-information can be used to extend the freshness interval or piggyback validations [15] to reduce the number of unnecessary `304 Not Modified` responses. We measure the potential applicability of this action by determining the number of bad performing sequences that contain any `304 Not Modified` response.

The last two server actions examine changing the way in which the content is delivered. The first approach is to reduce the amount of content by compressing it. We measure its applicability by determining the number of bad performing sequences with a large number of bytes. Secondly, we can alter content delivery by improving the server performance of handling many requests from a client through actions such as longer persistent connections or increased scheduling priority for this client at the server. Other techniques such as the bundling of content can also be used.

5. RESULTS

To study the client characterization policies and applicability of server actions, we collected a heterogeneous set of recent server logs from eight different Web server sites for the study. Descriptions for these Web sites are as follows:

- largesite.com—a large commercial site,
- research.com—a commercial research organization site,
- wpi.edu—an educational site for the WPI campus,
- cs.wpi.edu—a site for the WPI Computer Science Department,
- bonetumor.org—a medical site devoted to bone tumors, and
- cricinfo.org—a sports site devoted to cricket with logs from three mirror sites in Australia, U.K., and the U.S.A.

All logs are from the April–October, 2001 time frame except for the cricinfo.org logs, which are from June 2000. The duration, number of unique clients and number of requests for each log are shown in Table 4 along with the number of sequences we were able to identify using the methodology described in Section 4. The last column in the table indicates whether both the Referer and User-Agent fields were available for each log entry. We should note that we had additional sets of logs for these sites for similar durations. Although we show results for only one set, we did analyze the other sets and the results were similar to those reported here.

5.1 Client Characterization

Using these logs, we first studied the client characterization policies described in Section 4.2. While we studied all policy parameters described in this section for each of the logs shown in Table 4, we focus our discussion of results to particular policy parameters and logs for clarity. For example, all results shown use (5,8) i.e., the thresholds of $E_{first} > 5$ seconds or $E_{last} > 8$ seconds as definitions for identifying clients with poor performance. Using thresholds of (3,5) and (8,12) changes the number of poor performing clients identified in each log, but does not change the relative performance of the policies. Variations between other parameters and logs will be noted as appropriate.

5.1.1 Identifying Sequence Performance

Before studying the characterization of clients, we need to define a metric to measure the success of a classification policy. One such metric is to focus on the sequence accesses with "bad" and "good" performance. Consistent with our definition of E_{last}, we assume that a bad performing sequence is one where the delay for the last embedded object is greater than 8 seconds. Ideally all accesses with bad performance should come from a client who has been characterized as poor. In these cases the server could take a mitigating action to improve the performance in serving the given sequence of objects or altering the content served. Similarly, an access with good performance from a client characterized as poor could mean the server has taken some action, such as altering the page content, when in fact poor performance did not result. We assume an access with good

Table 3: Applicability of Server Actions to Sequences with Bad Performance

Server action	Nature of sequences to which to apply
Reduce the number of objects or bytes	Long sequence or large number total bytes
Redirect to replica or mirror	High first object delay
Alter meta-information	Presence of unnecessary 304 responses
Alter content delivery via compression	Large number total bytes
Alter content delivery by keeping sessions open longer	Long sequence

Table 4: Statistics About the Web Site Logs Used in the Study

Log	Duration	# of Unique Clients	# of Requests (Millions)	# of Page Sequences	Referer & User-Agent Fields Available?
largesite.com	1 wk	3385079	48.4	1210966	yes
research.com	1 mo	384679	5.2	250548	yes
wpi.edu	1 wk	59844	6.0	270749	no
cs.wpi.edu	1 mo	37433	1.0	63007	yes
bonetumor.org	7 mo	65397	1.7	143867	yes
aus.cricinfo.org	1 wk	25937	3.3	175496	no
uk.cricinfo.org	1 wk	76335	6.7	262254	no
usa.cricinfo.org	1 wk	38533	2.2	98908	no

performance shows a last embedded object delay of no more than 2 seconds.

As a baseline for measuring the success of classification, we first examine the percentage of bad performing sequences in each log. The second column in Table 5 shows the percentage of bad performing sequences where the delay for downloading the last embedded object is greater than 8 seconds. This value ranges from 9.2% in the cs.wpi.edu log to 38.5% in the USA cricinfo.org logs. Because it is difficult to classify clients until at least one sequence has been accessed by a client, the third column in Table 5 shows the percentage of bad performing sequences from a known client who has made at least one previous request. The final column in Table 5 shows the percentage of bad performing sequences from a client who has previously requested at least one sequence or is in a cluster for which another client in the cluster has previously requested a sequence. In cases such as largesite.com, the difference between the latter two columns indicates that client clustering can have a significant effect on the raising the potential ability to characterize clients.

Table 5: Pct. of Accesses with Bad Performance

Log	All Bad	From Known Clients	From Known Clusters
largesite.com	30.5	13.6	26.5
research.com	10.2	4.8	9.0
wpi.edu	14.7	10.9	13.4
cs.wpi.edu	9.2	3.8	4.4
bonetumor.org	29.3	16.9	25.4
aus.cricinfo.org	37.3	29.7	35.4
uk.cricinfo.org	35.2	26.5	33.8
usa.cricinfo.org	38.5	25.1	34.9

As a similar baseline, Table 6 shows the percentage of good performing sequences in each log. The second column in Table 6 shows the percentage of all good performing sequences where the delay for downloading the last embedded object is no more than 2 seconds. The respective columns are similarly defined as those in Table 5 and show that between 28.8% and 78.8% of all accesses show good performance. Clustering raises the percentage of good performing accesses coming from client clusters with repeat accesses.

Table 6: Pct. of Accesses with Good Performance

Log	All Good	From Known Clients	From Known Clusters
largesite.com	38.3	24.3	33.0
research.com	72.7	39.3	64.9
wpi.edu	67.5	62.2	65.5
cs.wpi.edu	78.8	53.8	60.9
bonetumor.org	47.9	27.1	41.1
aus.cricinfo.org	28.8	27.3	28.3
uk.cricinfo.org	34.3	27.6	32.4
usa.cricinfo.org	34.5	28.7	32.0

5.1.2 *Poor Client Characterization Policies*

Given the percentage of bad and good accesses in each log, the best policy for characterizing poor clients maximizes the number of bad accesses associated with poor clients while minimizing the number of good accesses by these clients. We looked at the range of policies summarized by the parameters in Table 2. For the initial discussion of our results we use the largesite.com logs.

The policy results are shown as labeled interior points in Figure 1. The "Best" point on the y-axis is taken from the second column of Table 5. For a policy seeking to associate bad performing sequences with clients classified as poor, this point represents the best a policy to classify poor clients could do. The "Worst" point on the x-axis is taken from the second column of Table 6. This point represents the worst a policy classifying poor clients could do.

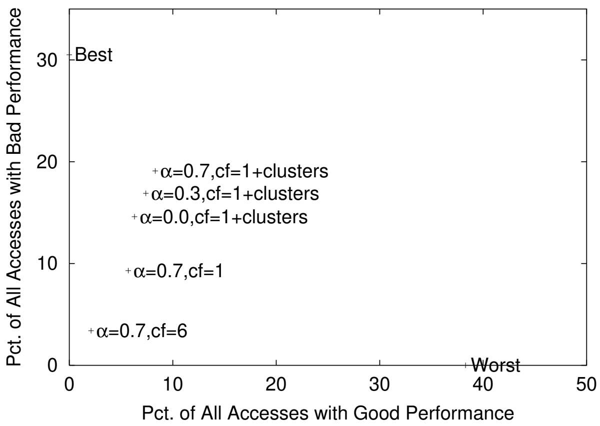

Figure 1: Accesses from Clients Classified as Poor for Selected Policies (largesite.com)

The interior points in Figure 1 represent results for particular policy parameters. The "α=0.7, cf=6" point in the lower left corner is a relatively conservative classification policy because it requires a confidence (cf) of at least 6 sequence accesses from a client before it is willing to classify the client as either poor or rich. Using this policy only 3.4% of all accesses are bad and come from a client classified as poor while 2.1% of all accesses are good and come from such clients. The "α=0.7, cf=1" policy classifies a client as soon as one sequence access from the client has been made. Using this policy only 9.3% of all accesses are bad and come from a client classified as poor while 5.7% of all accesses are good and come from such clients. Other α values for these confidence levels show similar results and are omitted from the graph for clarity.

The remaining three points on the graph show results where the characterization for all clients within a cluster is used to initialize the classification for any previously unseen client within a cluster. Results for all values of α with a confidence level of one access needed for classification are shown. These results show a much higher level of accesses that are bad and made by clients classified as poor, but also an increase in the number of accesses that are good by this set of clients. Using α=0.7, 19.1% of all accesses have a delay for the last embedded object greater than 8 seconds with these accesses done by a client characterized as poor. In these cases the server could potentially take an action, such as lowering the number of embedded objects, to reduce this time. On the other hand, using this same α value, 8.3% of all accesses have a delay of less than two seconds for the last embedded object and were made by a client classified as poor.

The intention of showing these different policy parameters is not to show which is best, but to show the range of tradeoffs that exist. The correct tradeoff depends on what is important to the site and what are the negative implications of an incorrect classification. If a site would like to retain clients who have poor connectivity to the server then the site might be more aggressive in classifying clients, particularly if the actions taken do not have an overtly negative effect on a client who might be misclassified. Otherwise, a more conservative policy should be considered. However, if the policy is so conservative that it classifies relatively few clients then the idea of client classification may not be useful at all for the site.

The results shown in Figure 1 do indicate that basing initial classification decisions on accesses from across a cluster of clients is worthwhile. More accesses with bad performance are associated with clients classified as poor while the ratio of bad to good performing sequences is about the same, or a bit better, than when clustering is not used.

5.1.3 Poor Client Characterization

Figure 1 shows results for different policy parameters for a single log. Table 7 shows results for all logs with for the policy parameter combination of α=0.7 and a needed confidence level of one access with clustering.

The second column is the percentage of accesses with bad performance from clients classified as poor with the relative percentage of all accesses with bad performance in parentheses. The results for the largesite.com and cricinfo.org sites show the best performance with more than 60% of all bad performing accesses from clients classified as poor. These are opportunities for the server to take mitigating action. The research.com, wpi.edu and cs.wpi.edu sites generally have better connected clients (many on-site users for the wpi logs) so classification of poor clients is expected to be less successful.

The last column in Table 7 reflects the potential negative effect of making an incorrect client classification. This column shows the percentage of accesses with good performance made by a client classified as poor. The relative percentage of these accesses compared to all good performing accesses (second column in Table 6) is given in parentheses. The results show 2.4–10.6% of all accesses fall in this category for with relative percentages from 3.0–36.1%.

Table 7: Accesses by Client Classified as Poor (α=0.7, cf=1 with clustering)

Log	Bad Performing Accesses (% All Bad)	Good Performing Accesses (% All Good)
largesite.com	19.1 (62.6)	8.3 (21.7)
research.com	3.4 (33.3)	4.5 (6.2)
wpi.edu	5.8 (39.5)	7.4 (11.0)
cs.wpi.edu	1.8 (19.6)	2.4 (3.0)
bonetumor.org	14.8 (50.5)	9.5 (19.8)
aus.cricinfo.org	25.7 (68.9)	10.4 (36.1)
uk.cricinfo.org	25.0 (71.0)	10.6 (30.9)
usa.cricinfo.org	26.3 (68.3)	9.8 (28.4)

5.1.4 Rich Client Characterization

As a contrast to characterizing poor clients, Table 8 shows results for characterizing rich clients for all logs with a policy parameter combination of α=0.7 and a needed confidence level of 6 accesses with clustering. A higher level of confidence is expected to classify a client as rich because the most likely server action to take in this case is serving enhanced content to the user. Doing so for a client incorrectly classified as rich could have serious negative effects for the client.

Table 8: Accesses by Clients Classified as Rich (α=0.7, cf=6 with clustering)

Log	Good Performing Accesses (% All Good)	Bad Performing Accesses (% All Bad)
largesite.com	3.5 (9.1)	0.3 (1.0)
research.com	19.0 (26.1)	0.6 (5.9)
wpi.edu	29.1 (43.1)	2.1 (14.3)
cs.wpi.edu	26.0 (33.0)	0.4 (4.3)
bonetumor.org	5.8 (12.1)	0.6 (2.0)
aus.cricinfo.org	2.4 (8.3)	0.3 (0.8)
uk.cricinfo.org	4.4 (12.8)	0.4 (1.1)
usa.cricinfo.org	3.7 (10.7)	0.3 (0.8)

The meaning of results shown in Table 8 mirror those shown in Table 7. The second column in Table 8 is the percentage of accesses with good performance from clients classified as rich with the relative percentage of all accesses with good performance in parentheses. The WPI and research.com results show the best absolute and relative performance. The last column of Table 8 shows the percentage of accesses with bad performance from clients classified as rich. As desired these absolute and relative percentages are low.

5.1.5 Per-Client Characterization

For a given policy, each client is initialized to the normal class unless access information is known for other clients within the same cluster in which case the client inherits the class of its cluster. Each client is potentially reclassified after each sequence access. For each policy, we counted the number of clients in each class at the end of processing each log. We also kept track of the class for clients who were inactive for over a day and whose state information was removed.

Table 9 shows the percentage of distinct clients in each class for the largesite.com logs. The policy parameters of α=0.7, cf=1 with clustering are used for the results. Clients are classified according to the number of sequence accesses they made in the log. Because client information is discarded after a day of inactivity, accesses from the same client separated by more than a day are classified as separate clients.

Table 9: Percentage of Distinct Clients In Each Class (largesite.com)

Number of Client Accesses	Client Class: Poor	Normal	Rich	Total
1-4	15.2	73.0	6.9	95.0
5-9	1.6	1.6	0.6	3.8
10-14	0.2	0.3	0.1	0.5
15-19	0.1	0.1	0.0	0.2
20-24	0.0	0.1	0.0	0.1
25+	0.1	0.2	0.0	0.3
Totals	17.2	75.2	7.6	100.0

The results show that the vast majority of clients coming to this site access a relatively few number of pages containing embedded objects. Among all clients, about 75% remain classified as normal. About twice as many of the remaining clients are classified as poor compared to rich. As a contrast, using the same policy parameters for the research.com logs shows 78% of clients classified as normal, 18% as rich as the remaining 4% as poor. The difference simply reflects different characteristics amongst the set of clients visiting these sites.

We also examined the client characterization policies for their stability in terms of the frequency that the client class changed. We found that for 72.5% of the largesite.com clients stayed in the same class as its initial class. In most cases, the client was initialized to normal and stayed in that class, but if cluster information was available then the client may have been initialized to the poor or rich class. For 18.6% of clients, there was at least one transition from the poor class to another class or from another class to the poor class. Given that 17.2% of all clients finished in the poor state (shown in Table 9), these results indicate that the classification is stable. In fact only 1.2% of clients had more than two transitions in or out of the poor class. Similarly, 9.8% of clients had at least one transition into or out of the rich class. Given that 7.6% of all clients finished in the rich class, these results also indicate stability of the classification. Only 1.1% of clients had more than two transitions in or out of the rich class.

5.2 Applicability of Potential Server Actions

In addition to using the Web server logs to classify clients, we also used the logs to identify sequences exhibiting bad performance and determine which of these actions could be applied to improve performance. For this portion of the study we analyzed the characteristics of all sequence accesses in which the delay for the last object retrieval was more than 8 seconds. We sought to match the sequence characteristics identified for possible server actions in Table 3 with the characteristics of accesses with bad performance in the set of logs. We considered all such accesses and did not limit our analysis to only those accesses from clients classified as poor.

The criteria to determine which sequences are appropriate for which server actions will vary according to the different administrative policies at each site. For purposes of comparisons between different types of sites and different server actions we apply a common criteria to all sites, but this approach is used only to understand the potential applicability of different server actions.

Table 10 shows the percentage of sequences with bad performance for which each type of server action is applicable. Table 5 has the results on the percentage of sequences with bad performance in each log.

We first examine the applicability of server actions that do not alter the contents of a page. The first of these actions is to redirect a client to a "closer" content provider. This action could be applied to any sequence, but we focus on sequences where the delay to retrieve the first embedded object in the sequence is large. A large initial delay may indicate the effect of a big RTT between the client and server, which can be reduced if the client can retrieve the content from a site that is closer to it. For these results we use a threshold of greater than 5 seconds because it is the corresponding threshold we used in classifying poor clients. The values in column two of Table 10 show the percentage

Table 10: Pct.of Sequences with Bad Performance for Application of Server Actions

Log	Redirect to Replica	Alter Meta-Information	Compress Content Delivery	Alter Object Delivery	Alter Page Contents
largesite.com	42	25	59	67	72
research.com	38	16	42	26	52
wpi.edu	26	43	54	56	71
cs.wpi.edu	31	17	50	29	68
bonetumor.org	37	10	83	3	84
aus.cricinfo.org	47	45	9	32	35
uk.cricinfo.org	59	39	10	20	23
usa.cricinfo.org	53	27	15	24	28

of all sequences with bad performance that have a delay for the first object greater than 5 seconds. The percentage of sequences is greater than 25% in all cases and shows the largest percentage across all potential server actions for the cricinfo.org logs.

Note that the initial delay would not include time spent by a server to generate the content because the timestamp on the base object is the time at which its content is written to the client. It is possible that a large base page could cause a longer initial delay as the client has more bytes to receive and process before it can request subsequent objects. For example in the largesite.com logs, 37% of all sequences with bad performance had a first object delay greater than 5 seconds and a base page larger than 10K bytes.

The next column in the table (altering meta-information) indicates the percentage of sequences that include at least one `304 Not Modified` response. These requests could be eliminated by increasing the freshness lifetime for objects or allowing clients to piggyback validation requests onto other requests. The results show that this approach could be of moderate usefulness, particularly for the wpi.edu and cricinfo.org logs.

It is difficult to apply a precise criterion for when compression is appropriate. We selected a value of 40K total bytes in a sequence as the threshold for a sequence with a large number of bytes. This is approximately the median value for total bytes in a sequence across all logs in our study. The medians for individual logs ranged from 15–65K bytes. The fourth column in the table shows the percentage of sequences with bad performance that have more than 40K total bytes. The bonetumor.org logs have the highest percentage of large sequences. All but the cricinfo.org logs have a significant number of sequences matching this criteria. A direction of future work is further distinction between text and image objects as image data cannot be generally compressed as much.

The next column in the table identifies the percentages of sequences with bad performance that have a long list of objects in the sequence. The handling of these long sequences can be altered by bundling objects together. Alternately a server can improve the effectiveness of handling each request by extending the connection lifetime or giving higher scheduling priority for these clients. The results shown in the table are the percentage of sequences with at least 10 objects. This is approximately the median length of sequences with poor performance across all logs. The medians for individual logs are between 4 and 14. The results show the most applicability of this server action for the largesite.com and wpi.edu logs. The bonetumor.org logs have virtually no long sequences.

In contrast to actions that do not alter page contents, the last column in the table shows the percentage of sequences with bad performance that could be improved by serving an altered (by size and number of objects) Web page. This action could be applied to all sequences, but is most applicable to those with either a large number of objects or total bytes. The percentages in the table are for all sequences with bad performance that have at least 10 objects or more than 40K total bytes. This action could be applied to the majority of sequences in all logs except for the cricinfo.org logs.

Additional work is needed to identify appropriate server actions for a sequence with bad performance. Beyond evaluating applicability we have to examine effectiveness. However, the results do indicate that each of the potential server actions could be applied to a significant number of sequences at least one site. In addition, the variety of server sites yield different server actions that would be most applicable.

5.3 Results Summary

There are number of results we wish to summarize about our basic premise that clients can be classified based on characteristics determined by a server and that a range of server actions could be applied in the case of a client classified as poor.

- A relatively simple classification policy can correctly classify clients as poor so that the majority of sequences with bad performance come from these clients.
- Client clustering improves the number of clients who can be initially classified based on cluster information in most logs and significantly improves in some logs. There are not only more classifications, but the accuracy of these classifications is comparable to when information for only a single client is used.
- There are tradeoffs for the aggressiveness of client classification. The negative implications of a wrong classification should determine the aggressiveness of the policy to use.
- Classification also helps characterize rich clients, but the impact is reduced when these policies need to use a greater degree of confidence because the cost of serving enhanced content to a client incorrectly classified as rich is higher.

- Client classification works best for sites with a variety of clients. Classification of poor clients does not work well when there are relatively small numbers of poor clients visiting the site.
- Client characterization is stable for particular clients with few transitions between classes observed beyond initial transitions for a client.
- Not a single type of server action is most applicable for all sites and each server action is applicable to a significant number of sequences with bad performance on at least one site.

6. RELATED WORK

The concept of dynamically altering multimedia content in a Web page depending on network path characteristics was reported in a recently issued United States patent [19]. In this proposed scheme, a Web server would monitor a variety of network characteristics (such as round trip time, packet loss) between itself and the client and correspondingly adjust the quality of the content returned to the client. However, this patent deals exclusively with altering the content or its delivery. It does not cover the range of other server actions that are part of our approach. Additionally, the network aware clustering we use to construct a coarse grouping of clients has been shown to be superior to the "nearness" approach discussed in [19]. We are not aware of any published research on the idea proposed in this patent.

Rather than let the server estimate a client's characteristics, other approaches can be used. Explicit client specification of network connectivity is used in many multimedia players. The SPAND system uses an approach where a group of clients estimate cost to retrieve resources and share it amongst themselves [23].

There is related work in adapting the content served to users based on server load [1]. As a server becomes loaded it begins to degrade the content served to lower-priority clients. In comparison with our work, server load is the only performance measure that triggers an action and the only action considered is to serve degraded content.

The Apache Web server's contributed collection of plug-ins include a plug-in called `Apache::Throttle` [24] that implements content negotiation based on the speed of the connection with the client. For example, poorly connected clients can receive lower resolution images. Connectivity is measured via a high resolution timer by noting the time between the beginning and the end of sending the response to the client at the application level.

Recent work also seeks to estimate the service time for a client through examination of server logs [4]. However this work focuses on how to more precisely estimate the total time by instrumenting an Apache server to log additional data. This information is used to analyze how the size of the write buffer at the server influences performance measurements. This work tries to make a more accurate estimation of client response time than our work, but uses this information for a much more narrow purpose.

7. SUMMARY AND FUTURE WORK

We have outlined a way to identify client's characteristics automatically by passive analysis of Web server logs. Even our simple set of categories (poor, normal, and rich) can be used to drive a wide range of possible server actions. The characterization does not have to be foolproof and our methodology allows more liberal and conservative classification approaches. We provide a map between classification and the kind of server action that might be most appropriate for a client. Although our classification is based on examining sequences of accesses, we are able to identify individual client's stability during the analysis: a more stable client stays in the identified category for a large portion of the analysis period. The stability of a client can help the server to decide if it is an appropriate candidate for tailoring server actions.

We have explored applicability of our methodology by conducting a range of experiments using a large number of real and extensive server logs. The logs were obtained from diverse organizations such as a university, a large corporation, and a popular sports site mirrored in multiple countries. We demonstrated stability of our client characterization technique in each of the logs. We estimated that significant portions of sequences with bad performance could benefit from some server action.

Our methodology is extensible; other techniques to characterize clients could be incorporated to increase confidence in the classification and server action. The use of network-aware clustering to coarsely group clients is also a helpful aid in the characterization. Our novel proposal can be tailored to an individual Web site, applied automatically, and validated to examine if the server actions result in specific improvements. The amount of state that needs to be maintained can be altered depending on the improvements.

While this work has examined the potential feasibility of client characterization and server adaptation, it leads to a number of directions for future work. First, we plan to examine a wider range of thresholds and policies for categorizing rich and poor clients as well as what constitutes good or bad performance. Second, we intend to validate our characterization results to see if poor/rich clients are really poor/rich by active measurements from known clients in different locations to a server under our control.

Third, we plan to instrument a real Web server to study the practicality of a server maintaining such information for clients and using the information to make decisions in serving content. We need to better understand the memory and computational overhead of our approach. Finally, we need to study not just the applicability of various server actions, but more importantly their effectiveness in reducing user-perceived latency.

Acknowledgments

We would like to thank all those who gave us access to their logs without which such research would be impossible. We thank Mikhail Mikhailov and the anonymous reviewers for their comments on an earlier version of the paper.

8. REFERENCES

[1] Tarek F. Abdelzaher and Nina Bhatti. Web Server QoS Management by Adaptive Content Delivery. In *Proceedings of the International Workshop on Quality of Service*, London, England, June 1999. http://www.eecs.umich.edu/~zaher/iwqos99.ps.

[2] Mark Allman. Measuring end-to-end bulk transfer capacity. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, November 2001.

[3] V. Almeida, D. Menasce, R. Riedi, F. Peligrinelli, R. Fonseca, and W. Meira Jr. Analyzing the impact of robots on performance of Web caching systems. In *Proceedings of the 6th International Web Caching Workshop and Content Delivery Workshop*, Boston, MA, June 2001.

[4] O. Ardaiz, F. Freitag, and L. Navarro. Estimating the time of service in a Web client starting from the server logs. In *Proceedings of the ACM SIGCOMM America Latina Conference*, San Jose, Costa Rica, April 2001. ACM.

[5] Pierre Beyssac. BING: Bandwidth pING, March 1998. http://www.cnam.fr/reseau/bing.html.

[6] Peter Bickford. Worth the wait? View Source, Human Interface Online, 1999. http://devedge.netscape.com/viewsource/bickford_wait.htm.

[7] Ramon Caceres, Fred Douglis, Anja Feldmann, Gideon Glass, and Michael Rabinovich. Web proxy caching: the devil is in the details. In *Workshop on Internet Server Performance*, Madison, Wisconsin USA, June 1998.

[8] CC/PP Working Group. http://www.w3.org/Mobile/CCPP/.

[9] Willy Chiu. Best practices for high volume web sites, February 2001. http://www.worldinternetcenter.com/Other_Events/Challenge-The-Expert/Feb28megawebsite/SVWIC_Chiu_2_28_01.pdf.

[10] Edith Cohen, Balachander Krishnamurthy, and Jennifer Rexford. Improving End-to-End Performance of the Web Using Server Volumes and Proxy Filters. In *Proceedings of the ACM SIGCOMM '98 Conference*, September 1998. http://www.acm.org/sigcomm/sigcomm98/tp/abs_20.html.

[11] Allen B. Downey. Using pathchar to estimate Internet link characteristics. In *Proceedings of the ACM SIGCOMM '99 Conference*, Cambridge, Massachusetts USA, September 1999. ACM.

[12] Balachander Krishnamurthy, Jeffrey C. Mogul, and David M. Kristol. Key Differences between HTTP/1.0 and HTTP/1.1. In *Proc. Eighth International World Wide Web Conference*, Toronto, May 1999. http://www.research.att.com/~bala/papers/h0vh1.html.

[13] Balachander Krishnamurthy and Jennifer Rexford. *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement.* Addison-Wesley, May 2001. ISBN 0-201-710889-0.

[14] Balachander Krishnamurthy and Jia Wang. On network-aware clustering of Web clients. In *Proceedings of ACM SIGCOMM*, August 2000. http://www.acm.org/sigcomm/sigcomm00/program.html.

[15] Balachander Krishnamurthy and Craig Wills. Study of Piggyback Cache Validation for Proxy Caches in the World Wide Web. In *USENIX Symposium on Internet Technology and Systems*, pages 1–12, December 1997. http://www.research.att.com/~bala/papers/pcv-usits97.ps.gz.

[16] Balachander Krishnamurthy and Craig E. Wills. Piggyback server invalidation for proxy cache coherency. In *Seventh International World Wide Web Conference*, pages 185–193, Brisbane, Australia, April 1998.

[17] Robert B. Miller. Response time in man-computer conversational transactions. In *Proc. Sprint Joint Computer Conference*, Montvale, NJ, 1968. AFIPS Press.

[18] J. Mogul, B. Krishnamurthy, F. Douglis, A. Feldmann, Y. Goland, A. van Hoff, and D. Hellerstein. Delta encoding in HTTP. RFC 3229, IETF, January 2002. Proposed Standard.

[19] Jeffrey C. Mogul and Lawrence S. Brakmo. Method for dynamically adjusting multimedia content of a Web page by a server in accordance to network path characteristics between client and server, June 2001. United States Patent 6,243,761.

[20] Jeffrey C. Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy. Potential benefits of delta encoding and data compression for HTTP. In *Proceedings of the ACM SIGCOMM 1997 Conference*, pages 181–194, August 1997. http://www.acm.org/sigcomm/sigcomm97/papers/p156.html.

[21] Jakob Nielsen. *Designing Web Usability*. New Riders, 2000.

[22] Ram Rajamony and Mootaz Elnozahy. Measuring client-perceived response time on the WWW. In *USENIX Symposium on Internet Technology and Systems*, San Francisco, California, USA, March 2001.

[23] Srinivasan Seshan, Mark Stemm, and Randy H. Katz. SPAND: shared passive network performance discovery. In *USENIX Symposium on Internet Technologies and Systems*, Monterey, California, USA, December 1997.

[24] Apache::Throttle - Apache/Perl module for speed-based content negotiation. http://www.dmi.usherb.ca/laboratoires/documentations-logiciels/Perl/lib/Apache/Throttle.html.

[25] Craig E. Wills, Mikhail Mikhailov, and Hao Shang. N for the price of 1: Bundling Web objects for more efficient content delivery. In *Proceedings of the Tenth International World Wide Web Conference*, Hong Kong, May 2001. http://www.cs.wpi.edu/~cew/papers/www01.pdf.