Effective Web Browsing Through Content Delivery Adaptation

KANAME HARUMOTO
Osaka University
TADASHI NAKANO
University of California, Irvine
and
SHINYA FUKUMURA, SHINJI SHIMOJO, and SHOJIRO NISHIO
Osaka University

This article presents a Web content adaptation and delivery mechanism based on application-level quality of service (QoS) policies. To realize effective Web content delivery for users, two kinds of application-level QoS policies, transmission time and transmission order of inline objects, are introduced. Next, we define a language to specify these policies. We show that transmission order control can be implemented using HTTP/1.1 pipelined requests in which a client recognizes the transmission order description in a Web page and simulates parallel transmission of inline objects by HTTP/1.1 range requests. Experimental results show that our proposed mechanism realizes effective content delivery to a diverse group of Internet users. Finally, we introduce two methods to specify application-level QoS policies, one by content authors, and the other by end users.

Categories and Subject Descriptors: H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation (efficiency and effectiveness)*

General Terms: Design, Languages, Performance

Additional Key Words and Phrases: Content adaptation, hypertext, World Wide Web

1. INTRODUCTION

Currently, the World Wide Web is being used extensively as an important medium for information dissemination. Its high costeffectiveness is prompting

This work was supported in part by The 21st Century Center of Excellence Program of the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan, and in part by the Research and Development Program of Ubiquitous Network Authentication and Agent (2003), the Ministry of Internal Affairs and Communications (MIC), Japan.
Authors' address: K. Harumoto, Graduate School of Engineering, Osaka University, 2-1 Yamadaoka, Suita, Osaka 565-0871, Japan; email: harumoto@eng.osaka-u.ac.jp.

more and more companies to go online and engage in such types of e-business as Internet shopping, auctions, advertisements, and E-trade.

Web designers must take several conditions into consideration to effectively provide Web content to users. First, although Internet bandwidth is becoming increasingly broader, content transmission speed from Web servers and end users is becoming increasingly diverse. This is because a variety of connection methods have become available such as dial-up, ADSL, and wireless. According to the report of the Ministry of Public Management, Home Affairs, Posts and Telecommunications in Japan, over 70% of home users are using narrow-band dial-up connections as of 2002. Therefore, not all users equally experience smooth Web access, and users often have to wait an inordinately long time to download a Web page which can cause frustration. Such a long download time is referred to as the World Wide Wait, and estimates in a recently published report indicate that the e-business market has lost 21 billion dollars per year due to impatient users who have terminated their Web transfers. This situation will not disappear even if broadband wireless access becomes widely available because the diversity of network access methods will remain, and Web pages will get "fatter" (i.e., more high-quality and high-resolution images will be included) as network bandwidths broaden.

Second, as the WWW becomes a popular tool for getting information, devices other than personal computers, such as PDAs and cellular phones, are emerging that can access the Web. These devices have different characteristics with regard to display size, color depth, and acceptable content description languages.

Third, users have different preferences with regard to content delivery. For example, one user may want to download a Web page within 10 seconds, while another user may tolerate longer download times to get high-quality images.

These situations make it difficult for Web designers to design appropriate Web pages for every user. Since it is costly to prepare and maintain a variety of Web pages to meet the diverse demands of each Internet user, most Web designers basically provide only a single set of Web pages—usually visually-appealing Web pages with a large number of images, videos, and sounds to attract the attention of users. This is especially true for e-business sites that maintain a large number of Web pages.

To cope with these conflicting conditions, many efforts have been made in content adaptation [Fox and Brewer 1996; Fox et al. 1998a, 1998b; Shimada et al. 1997; Han et al. 1998]. One representative effort is content transcoding that automatically converts content description languages (e.g., from HTML to WML) and image formats (e.g., from JPEG to GIF) and then adjusts the quality of inline objects in a Web page according to a given client's network bandwidth and/or device capability. By adjusting the quality of multimedia content according to a client's network bandwidth, page download time can be reduced. However, the existing content transcoding mechanisms lack flexibility. For example, no mechanism exists that allows the content author to explicitly specify an upper limit for the transmission time, the most crucial parameter in overcoming the World Wide Wait. Moreover, most of these

approaches do not consider an important part of the nature of Web pages, that is, they use streaming just as video and audio streams do. The streaming nature of Web pages offers another possibility for reducing user-perceived latency. That is, transmitting inline objects in an appropriate order has the potential to reduce user-perceived latency and to improve the availability of Web pages.

In this article, we describe the design of an adaptive content delivery mechanism that enables a single set of Web pages to adapt to a wide range of network connections, from low-speed to high-speed ones. The adaptation in our mechanism is performed based on a set of policies that we call *application-level QoS policies* or *a-policies*. We can explicitly specify these a-policies in terms of transmission time threshold, quality prioritization, format conversion conditions, and transmission order; which allows us to keep the quality of a Web page as high as possible. Thus, our contributions in this article are to define a standard language to specify various kinds of policies and to provide a framework to guarantee the transmission of adapted Web pages based on these policies. Our framework enables a single set of Web pages to adapt to a wide range of networks which we believe will free content authors from the burden of preparing multiple versions of the same Web page.

In addition, we introduce two kinds of methods to specify a-policies, one by content authors, and the other by end users. Content authors might consider the effort to describe a-policies a heavy burden, especially at commercial sites in which a large number of Web pages have to be managed. Thus, we provide an authoring tool equipped with an a-policy editor and generator to automatically create a-policies based on created templates. These templates contain semantic information expressing how the author wants page(s) to appear for a client (for example, display product images before advertisement images). By applying the template to Web pages, the tool automatically generates a set of a-policies, providing content authors an efficient and generic way to specify a-policies. On the other hand, appropriate a-policies should depend not only on content authors, but also on the Web page viewer as well which motivates us to develop another specification method. With this method, we try to reflect a user's preferences for the kinds of a-policies that should be created by introducing preference profiles where an adaptation policy from the user's point of view can be described.

The rest of the article is organized as follows. In Section 2, we define application-level QoS policies and show examples of QoS policy descriptions. Section 3 presents a protocol for guaranteeing one of the a-policies, that is, the transmission order control policy. In Section 4, we explain the implementation of our adaptive content delivery system which utilizes a media-scaling mechanism to guarantee Web page transmission within a specified time. Section 5 shows evaluation results from efforts to guarantee a-policies such as transmission time and transmission order control. In Section 6, we briefly introduce the methods used to specify a-policies by content authors and by end users. We outline related work in Section 7. Finally, Section 8 concludes this article and describes the directions of our future work.

2. DESCRIPTION OF APPLICATION-LEVEL QoS POLICY

In this section, we introduce a language for content authors to specify application-level QoS policies (a-policies for short) for Web content adaptation. Although there are a variety of a-policies to consider, two types are essential namely, transmission time and transmission order. Section 2.1 describes the motivation for studying each policy, and section 2.2 shows examples of policy specification and description.

A description of a-policies is embedded in an HTML document and interpreted by a client before issuing requests for inline objects. The client regards the embedded a-policies as the content author's intentions and issues requests for inline objects according to those a-policies. The server also uses the a-policies to perform content adaptation. Note that there is another method to derive a-policies which will be presented in Section 6.

2.1 Motivation

2.1.1 *Transmission Time.* As stated in the previous section, transmission time is a very important factor in downloading Web pages. To prevent users from becoming irritated due to a long download time and intentionally terminating the delivery of a Web page, the page transmission time must be controlled. Therefore, content authors need to specify an upper limit for the transmission time (transmission time threshold) for each Web page as an a-policy.

Here we have to consider a few points. First, in order to deliver the Web page within the specified transmission time threshold, the quality of inline objects, especially of still images, may have to be reduced. However, there are situations where some inline objects are important and their quality needs to be kept as high as possible, while others are not so important. For example, in a company's product lineup page, the photo images of new products should be given a higher priority when compared with the images of other products. Therefore, content authors should be allowed to specify the priority of each image. This quality prioritization allows us to keep the overall quality of the Web page as high as possible.

Our second consideration concerns image formats. The image formats commonly used in the WWW are JPEG and GIF. We can control the quality of JPEG images by means of a parameter called quality factor with which we can reduce the data size of the image significantly while maintaining image quality. However, we cannot control the quality of GIF images in such a way. To reduce the data size of a GIF image, we must reduce the number of colors used or reduce the resolution of the image and, unfortunately, these conversions may result in a low-quality image and may cause the loss of image semantics. Therefore, content authors should be allowed to specify a format conversion from a GIF to a JPEG image so that a GIF image's quality can be acceptably controlled; however, some of the characteristics of the GIF image such as transparent color allocation and animation may be lost in the conversion process.

Based on this specification, the content delivery mechanism should adjust the quality of the inline images so that the page is transmitted within the specified threshold time.

2.1.2 *Transmission Order.* The transmission order of inline objects is especially important for users accessing the Internet with a low-speed link such as a dial-up line since inline objects in a Web page are generally presented on a Web browser in the same order they are transmitted. Therefore, content authors should be given a framework in which to specify the transmission order of inline objects. We consider three types of transmission order: sequential transmission, parallel transmission, and partial transmission and now give an explanation for each of them.

2.1.2.1 *Sequential Transmission.* Transmitting inline objects that are more interesting to users before other inline objects might reduce user-perceived latency. For example, in a company's product lineup page, most users are interested in information on new products. Therefore, transmitting new product images prior to old product images would be effective. In the top page of a Web site, transmitting images for links first is also effective because users can immediately follow the links.

2.1.2.2 *Parallel Transmission.* In recent Web pages, one large picture is often divided into multiple small images in order to assign each region to a different link or to create visual effects based on the behavior of the user's mouse pointer. In such cases, transmitting the images in parallel is effective because users can get a quick preview of the whole picture.

2.1.2.3 *Partial Transmission.* In the case where a Web page first delivers a largesized image, such as an animated image, the other images will appear inevitably late on the Web browser. To address this, we can partially transmit the image (say, only the first frame of an animated image) first, and then transmit the remaining frames after transmitting the other images. This partial transmission is effective because presenting the first frame is adequate enough for displaying the appearance of the Web page. This idea is not only applicable to animated images, but also to progressive JPEG and interlaced GIF images.

2.2 Specification and Description Example

In our framework, a-policies for a Web page are specified within that Web page's HTML code. This enables a browser-conducted content delivery control system (see Section 3).

2.2.1 *Transmission Time Threshold.* Transmission time threshold is specified by a `meta` element in the `head` section of an HTML document. The following states that a content author prefers to transmit a Web page within 30 seconds.

```
<head>
  <meta name="TransmissionTime" content="30">
</head>
```

2.2.2 *Quality Prioritization.* The quality of an image is specified by the `priority` attribute in an `img` element. In the following, the quality of *foo.jpg* is preserved at a higher level than that of *bar.jpg*.

```
<!ELEMENT dto  (par|seq)*>
<!ATTLIST dto  rest   (par|seq)           "seq"
               cache  (immediate|ordered) "immediate">
<!ELEMENT seq  (par|xfer)+>
<!ELEMENT par  (seq|xfer)+>
<!ELEMENT xfer EMPTY>
<!ATTLIST xfer object IDREF #REQUIRED
               range  CDATA #IMPLIED
               volume CDATA #IMPLIED>
```

Fig. 1. Document type definition for transmission order description.

```
<img src="foo.jpg" priority="high">
<img src="bar.jpg" priority="low">
```

2.2.3 *Format Conversion Condition.* The format conversion condition is specified by the `allow` attribute in an `img` element. The following specification indicates that a format conversion from GIF to JPEG will occur should the image quality become too low.

```
<img src="foo.gif" allow="ToJpegIfLowQuality">
```

As for animated GIF images, the content author can specify quality control by reducing the number of frames with the following method.

```
<img src="bar.gif" allow="ReduceFrameRate">
```

To apply these specifications to every image of the page, the content author describes the following in the `head` section.

```
<head>
  <meta name="Allow" content="ToJpegIfLowQuality">
</head>
```

2.2.4 *Transmission Order.* To specify the transmission order of inline objects, we have designed a transmission order description language based on XML which is the standard format for metadata on the WWW. Figure 1 shows the DTD (Document Type Definition) of the transmission order description language. The transmission order is specified by four elements: `seq` (sequential transmission), `par` (parallel transmission), `xfer` (transfer of an inline object) and `dto` (description of transmission order). We describe the syntax of these elements in the following. Note that each inline object whose transmission order is specified must be assigned a unique identifier in the `body` section of an HTML document, while content authors describe the transmission order in the `head` section with reference to the assigned unique identifiers.

(1) *The* `seq` *element.* A `seq` element indicates sequential transmission of inline objects. Inline objects indicated by the elements enclosed between the start and end tags of the `seq` element are transmitted in sequential order. A `seq` element can contain one or more `par` elements and `xfer` elements.

(2) *The* `par` *element.* A `par` element indicates parallel transmission of inline objects. Inline objects indicated by the elements enclosed between the start

and end tags of the par element are transmitted in parallel. A par element can contain one or more seq elements and xfer elements.

(3) *The* xfer *element.* An xfer element indicates transfer of the object assigned to the object attribute. To specify a partial transmission, a volume or range attribute can be used. The value of the volume attribute indicates the amount of the fragment to be transferred (e.g., volume="3000b"), while the value of the range attribute indicates the range of the fragment to be transferred (e.g., range="0b-2999b"). In both cases, the unit of the value can be specified by bytes (b) or percentages (%), and if the target object is an animated image, the frame unit (f) can also be used. In addition, specifying the value as rest indicates the transfer of all remaining parts of an object. Note that content authors must use the same attribute (volume or range) and the same unit (b, %, or f) consistently for one object in the transmission order description. As for partial transmission, content authors are prohibited from specifying a volume value larger than the actual data size of the object and from specifying an invalid range value. In addition, if all parts of an object are not specified, its remaining parts are transferred after the ordered objects are transferred.

(4) *The* dto *element.* A dto element encloses the transmission order description. This element can have rest and cache attributes. The rest attribute is used to specify the transmission order of inline objects whose order is not explicitly specified. Those inline objects will be transmitted after the transmission of the inline objects whose order is explicitly specified either in sequential order or in parallel, according to the value of the attribute. The possible values of this attribute are seq and par. On the other hand, if a client has some cached copies of the inline objects, there are two ways to present them on the browser: presenting those cached objects according to the transmission order description or presenting them immediately without waiting for other noncached objects to be transferred. With the cache attribute, content authors control the presentation order of cached copies. The possible values of this attribute are ordered and immediate.

Example 2.1. We present an example of the transmission order description, using the company Web page illustrated in Figure 2. This page includes various inline objects: an image of the company's logo; a new product picture that actually consists of four still images (X-1, X-2, X-3 and X-4) and one animated image (X-5); six image links to navigate to other pages such as the "what's new" page and the "product lineup" page; a background image; some background sound effects; and other miscellaneous information. We suppose that the site owner places the following demands on the transmission of this page.

(1) S/he wants to deliver this page to any user within 10 seconds to avoid an impatient user terminating the download.
(2) S/he thinks that the company's logo is very important and wants to preserve its quality in comparison with other objects.
(3) All images are in GIF format, but s/he will allow conversions to JPEG.

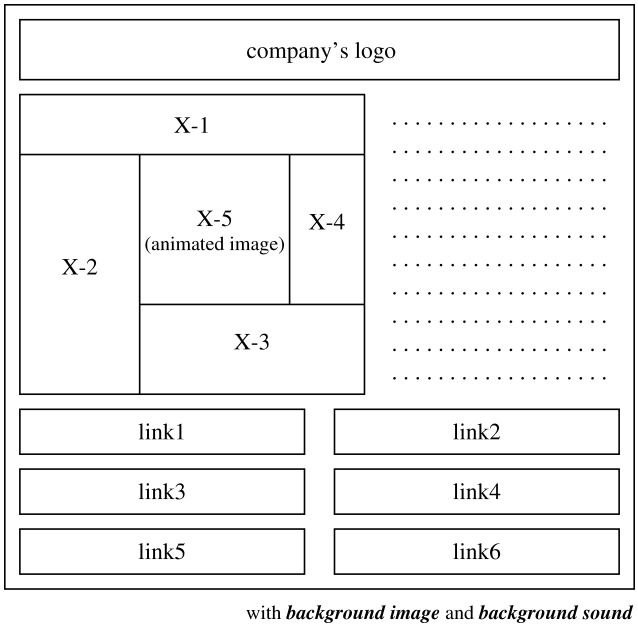


Fig. 2. Sample Web page of a company.

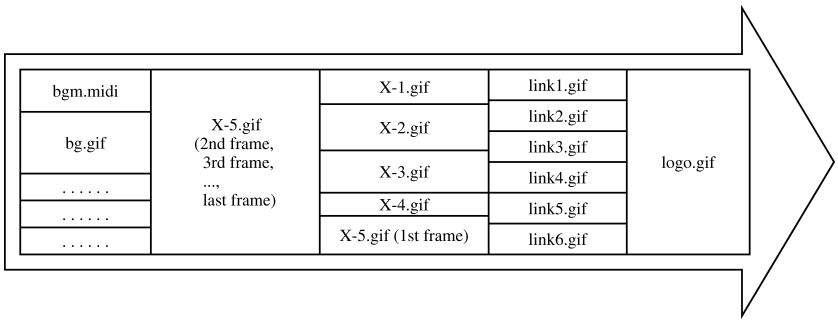


Fig. 3. Transmission order for the Web page illustrated in Figure 2.

(4) S/he wants to control the transmission order of inline objects. To promote the company's name, the logo image should be transmitted first. To provide users with quick access to other Web pages, link images should be transmitted next. Since the new product picture consists of five images, they are transmitted in parallel so that the user can get a quick preview of the whole picture. The animated image should be transmitted as a partial transmission to prevent the other four surrounding images from being presented too late on the client browser.

Figure 3 depicts this transmission order, and Figure 4 shows the a-policy description corresponding to the demands of the site owner.

```
<html>
<head>
  <title>Company's top page</title>
  <meta name="TransmissionTime" content="10">
  <meta name="Allow" content="ToJpegIfLowQuality">
  <dto rest="par" cache="ordered">
    <seq>
      <xfer object="logo"/>
      <par>
        <xfer object="l1"/>
        <xfer object="l2"/>
        <xfer object="l3"/>
        <xfer object="l4"/>
        <xfer object="l5"/>
        <xfer object="l6"/>
      </par>
      <par>
        <xfer object="X1"/>
        <xfer object="X2"/>
        <xfer object="X3"/>
        <xfer object="X4"/>
        <xfer object="X5" volume="1f"/>
      </par>
      <xfer object="X5" volume="rest"/>
    </seq>
  </dto>
</head>
<body background="bg.gif">
  <embed src="bgm.midi" autostart="true" loop="true">
  ...
  <img src="logo.gif" priority="high" id="logo">
  ...
  <table cellspacing="0" cellpadding="0">
   <tr>
    <td colspan="3"><img src="X-1.gif" id="X1"></td>
   </tr>
   <tr>
    <td rowspan="2"><img src="X-2.gif" id="X2"></td>
    <td><img src="X-5.gif" id="X5"></td>
    <td><img src="X-4.gif" id="X4"></td>
   </tr>
   <tr>
    <td colspan="2"><img src="X-3.gif" id="X3"></td>
   </tr>
  </table>
  ...
  <table cellspacing="0" cellpadding="0">
   <tr>
    <td><img src="link1.gif" id="l1"></td>
    <td><img src="link2.gif" id="l2"></td>
   </tr>
   <tr>
    <td><img src="link3.gif" id="l3"></td>
    <td><img src="link4.gif" id="l4"></td>
   </tr>
   <tr>
    <td><img src="link5.gif" id="l5"></td>
    <td><img src="link6.gif" id="l6"></td>
   </tr>
  </table>
  ...
</body>
</html>
```

Fig. 4. Example of a-policy description.

3. A PROTOCOL TO CONTROL TRANSMISSION ORDER

In this section, we discuss how we can implement transmission order control using the standard HTTP/1.1 protocol. In Section 3.1, we introduce a basic idea called transmission serialization to realize sequential and parallel transmission on a single TCP connection. Next, we describe how transmission serialization can be implemented with HTTP protocol in Section 3.2.

3.1 Serialization on a TCP Connection

In Frystyk et al. [1997], it was shown that a properly implemented HTTP/1.1 [RFC2616 1999] will often outperform HTTP/1.0 and reduce Internet traffic due to the introduction of persistent connections and pipelining which allow a client to make multiple requests without waiting for each response on a single TCP connection. Furthermore, since it is difficult to synchronize transmissions on multiple TCP connections, we have decided to use a single TCP connection to implement our transmission order control.

To transfer multiple inline objects over a single TCP connection in a specified order, we must rearrange the bytes of the inline objects. We call such rearrangement transmission serialization and define the arranged transmission as serialized transmission. Transmission serialization is performed as follows.

—Sequential transmission is realized by simply aligning the contained objects in a specified order.

—Parallel transmission is emulated by subdividing every object contained in the parallel transmission into segments of fixed-size (S) and aligning them by the following procedure:

(1) The first segment of each object is aligned first.
(2) The next segment of the object that has the smallest ratio of the total size of already aligned segments to its object size is aligned next.
(3) Repeat step 2 until all segments are aligned.

When the transmission order is specified in a combination of sequential and parallel transmissions, it can be implemented in a similar way.

Example 3.1. Suppose that the following transmission order is specified.

```
<dto>
  <par>
    <xfer object="A"/>
    <xfer object="B"/>
    <seq>
      <xfer object="C"/>
      <xfer object="D"/>
    </seq>
  </par>
</dto>
```

When the delivery mechanism delivers inline objects A, B, C, and D, the transmission serialization is performed as illustrated in Figure 5.

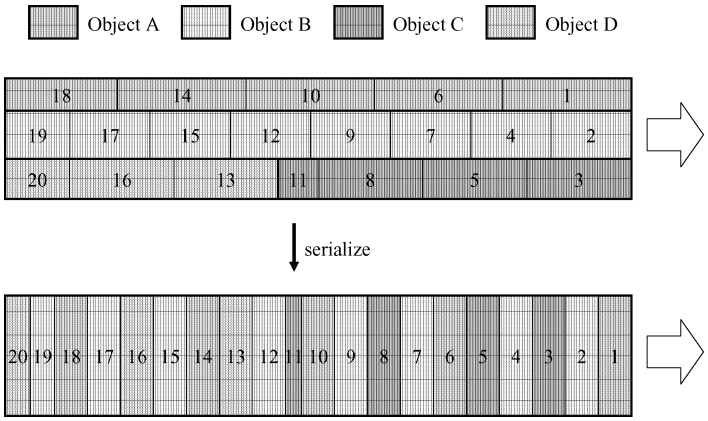


Fig. 5. Transmission serialization.

3.2 Transmission Order Control with HTTP/1.1 Range Requests

To implement the serialized transmission of inline objects, we can employ HTTP/1.1 range requests. A range request is a GET request with a Range header indicating a byte range to be retrieved. The range request is intended to reduce unnecessary network usage by allowing partially retrieved objects to be completed without transferring data already held by the client. Here, by forcing a client to issue the range request for each subdivided segment in Figure 5, we can implement serialized transmissions.

The interaction between the Web browser and the server is as follows. First, the browser sends a request for an HTML document to the server as usual and receives a response. Then the browser parses the received document to extract the description of the transmission order of inline objects. If a transmission order description exists, the browser performs the transmission serialization and sends requests according to it. If the transmission order description includes a parallel transmission, the browser must know the file sizes of the inline objects contained in the parallel transmission to serialize the transmission. This can be done by examining the Content-Range header in response to the request for the first segment of each object. Then, the browser can perform the transmission serialization for the parallel transmission and send requests for segments in the serialized order using range requests.

In this way, when we employ HTTP/1.1 range requests, there is no need to change the server, and only the Web browser need be implemented to understand the specification of the transmission order and to issue range requests to the server.

Although this transmission order control procedure generates more requests and responses than the case where transmission order is not controlled, the overhead should be small because the request and response messages can be pipelined in a single connection. The overhead is evaluated in Section 5.2.

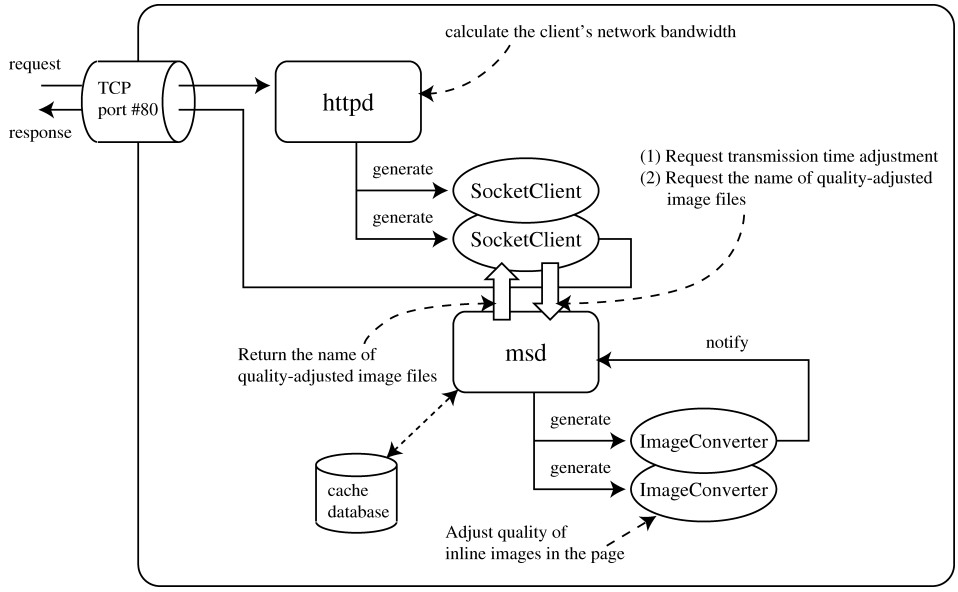


Fig. 6. System implementation.

4. IMPLEMENTATION DESIGN

4.1 Architecture

Figure 6 shows the system architecture of the proposed content delivery mechanism. The system consists of two modules: httpd (HTTP daemon) and msd (media-scaling daemon). The httpd is an extended HTTP daemon with the following functions: accepting connection requests from clients, monitoring network bandwidth between the server and the clients, and requesting the msd to adjust image quality if necessary. The msd is responsible for adjusting the quality of inline images in the requested Web page according to the specifications described in the page. These modules work as follows.

(1) A client sends a request for an HTML file to the httpd running on the server. The httpd must identify a series of requests from a client so that quality adjustment of inline images can be properly performed. Assuming the existence of proxies or a NAT mechanism between the client and the httpd server, we cannot use the IP address of the request source. Instead, we use a session cookie to identify a client.

(2) The httpd returns the requested HTML file which may contain the transmission order description. During the transmission, the httpd monitors the network bandwidth between the server and the client by keeping track of the elapsed time and amount of bytes transmitted. Then, using the obtained network bandwidth information as a parameter, the httpd issues a request to the msd for quality adjustment of the inline images in the HTML file.

Note that because the network monitoring is performed at HTTP level (not at network level), the amount of bytes does not include bytes of TCP/IP or

Original file name	Adjusted file name	Size	Parameter	Entity tag
/foo.jpg	/foo.jpg	20074	*original*	e91c3-4e6a-40559960
/foo.jpg	/adjusted/foo-75.jpg	16323	quality=75	e95eb-3fc3-40ecad1a
/foo.jpg	/adjusted/foo-70.jpg	12863	quality=70	e95e7-323f-40ecabc6
/foo.jpg	/adjusted/foo-50.jpg	9445	quality=50	e95e8-24e5-40ecabf7
/foo.jpg	/adjusted/foo-30.jpg	6878	quality=30	e95ea-1ade-40ecac2d

Fig. 7. Example of cache database.

other network protocol headers, and the calculated bandwidth differs from the effective bandwidth as in the original meaning. The httpd monitors the effective bandwidth at the application level.

(3) The msd parses the HTML files to extract the file names of the embedded images and the specifications such as transmission time threshold. Next, the msd calculates the reduction rate R from the total amount of data to be transmitted (D), the specified transmission time threshold (T), and the network bandwidth (B) given by the httpd:

$$R = \frac{B \times T}{D}.$$

If $R < 1$, the msd tries to generate a quality-adjusted image file for each original image. Before quality adjustment of each image, the msd checks whether there exists an appropriate quality-adjusted version of the image in the cache directory, using the cache database. Figure 7 shows an example of the cache database. If the msd finds the appropriate quality-adjusted version of the image, the quality adjustment process is skipped. Otherwise, the msd reduces the file size of the image by controlling its quality so that the file size becomes about R times the file size of the original image. The msd stores the quality-adjusted image in the cache directory and records its information on the cache database.

Note that if the image is prioritized, the msd adjusts its R in this phase. In addition, the msd might convert an image format according to the format conversion conditions described if necessary.

(4) The client also parses the received HTML file. If the client is equipped with the transmission order control feature, the client recognizes the transmission order description and issues requests for inline images to the httpd according to the transmission serialization (see Section 3.2). Otherwise, the client ignores the transmission order description and issues requests for inline images as usual.

If the client has a client cache of an image to be requested, an If-None-Match header with the entity tag of the client cache is included in the request to validate the client cache.

(5) When the httpd receives a request for an image from the client, it makes an inquiry to the msd to obtain the file name of the quality-adjusted image. If the request contains an If-None-Match header, the httpd includes the entity tag indicated by the header in the inquiry. The client cache is judged to be valid by the msd if the following three conditions hold: the entity tag is found in the cache database, the corresponding image file has been

generated from the current original image, and its data size is large enough considering the reduction rate R. In this case, the msd notifies the httpd that the client cache is valid. Otherwise, the msd returns the name of the corresponding quality-adjusted image file. If the quality adjustment has not been completed yet, the msd returns the file names as soon as the quality adjustment is completed.

(6) When the httpd is notified that the client cache is valid, it returns a status code 304 (Not Modified) to the client. Otherwise, the httpd returns the content of the possibly quality-adjusted image file indicated by the msd, with a status code 200 (OK) or 206 (Partial Content), as if the content is of the original image. The httpd includes the entity tag of the quality-adjusted image file in the response for subsequent cache validation.

Note that if the client uses range requests to perform the parallel transmission of inline images, there is a possibility that a requested image file will be modified during the range requests. However, the client can detect such a modification by checking the entity tags attached to the responses. When such a modification is detected, the client issues an additional range request to recover the part of the image file that has already been received.

During the transmission, the httpd also monitors the network bandwidth. It keeps track of the amount of transmitted bytes and elapsed time and uses this bandwidth information when it processes succeeding page requests from the same client.

4.2 Implementation Details

4.2.1 *Httpd.* Our implementation of the httpd was based on the Jigsaw WWW server [W3C-Jigsaw]. When the httpd receives a connection request from a client, it generates SocketClient in a thread. The SocketClient exchanges data between the server and the client. It also monitors the effective network bandwidth. Note that the effective bandwidth is difficult to estimate when the client enjoys data compression such as a PPP (Point-to-Point Protocol) connection and V.42bis on a modem since the compression rates of text data and image data are considerably different. Taking this fact into account, the httpd records the bandwidth information separately for text data (i.e., HTML files) and binary data (i.e., image files). Another issue related to bandwidth estimation is that little bandwidth information is available for clients who have only just recently started requesting pages, and we are obliged to use bandwidth information obtained through the transmission of the first-requested HTML file for quality adjustment. Unfortunately, this may cause an estimation error as shown in Section 5.1.

4.2.2 *Msd.* We implemented the msd in Java. The msd accepts two types of messages: a request for quality adjustment and an inquiry for the name of the file to be transmitted. If the message is a request for quality adjustment, the msd generates a thread that performs actual quality adjustment with the ImageConverter.

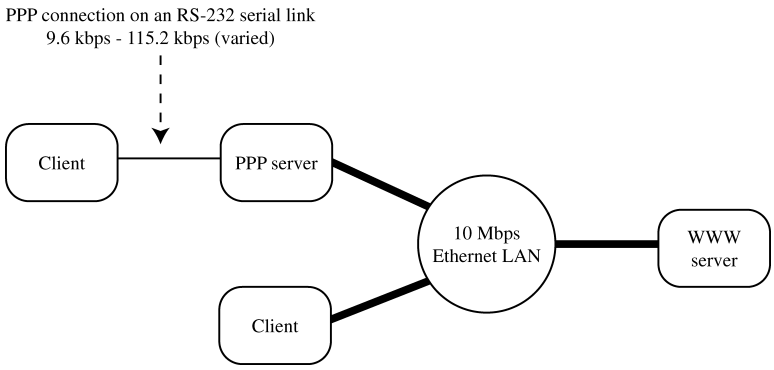


Fig. 8. Experimental setup.

In order to adjust the quality of images, some parameters such as the quality factor for JPEG images are adjusted. However, estimating a priori an accurate data size of the adjusted image file is difficult. Furthermore, there are cases where we cannot reduce the data size to the target data size. Therefore, we allow a tolerance of 10% for the data size and limit the number of trials of quality adjustment to three. We also set the lower limit of the reduction rate (R) to 0.25 to keep the images recognizable. The quality-adjusted images are stored in the cache directory and their data sizes, adjustment parameter values, and entity tags are recorded on the cache database. This helps to reduce the cost of quality adjustment.

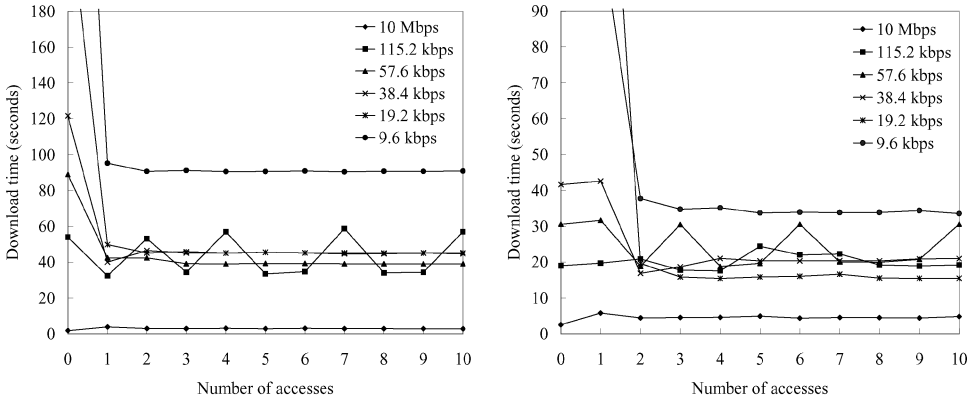
5. EVALUATION

In this section, we show some evaluation results for the implemented content delivery mechanism. Section 5.1 examines our system's ability to guarantee a transmission time threshold. Section 5.2 evaluates the performance of transmission order control protocol using HTTP range requests compared with standard HTTP operation. Section 5.3 shows the actual impact of a transmission order control on user-perceived latency by using two typical Web pages.

5.1 Transmission Time Guarantee

This experiment examines how well the content delivery mechanism can estimate the available bandwidth and carry out a Web page transmission within a specified transmission time threshold. Figure 8 depicts the experimental setup used to evaluate the implemented WWW server. To evaluate our WWW server at a variety of connection speeds, we set up two clients; one was connected directly to the LAN via a 10Base-T Ethernet, while the other was connected to a PPP server via an RS-232 serial link. We varied the serial port speed between 9.6 kbps to 115.2 kbps to simulate low-speed connections. In this environment, we measured the download times for a set of Web pages. Note that client caching is disabled in this experiment. Here we show two representative sets of results.

Figures 9(a) and 9(b) show the download times to access pages A and B of Table I, respectively. In these graphs, the horizontal axis indicates the number



(a) Page A (Threshold = 20 seconds). (b) Page B (Threshold = 40 seconds).

Fig. 9. Page download time.

Table I. Page Configuration and Specified Transmission Time Threshold

Page	HTML	Images	Total Size	Transmission Time Threshold
A	7.2 KB	170.4 KB (20 images)	177.6 KB	20 seconds
B	2.0 KB	447.0 KB (11 images)	449.0 KB	40 seconds

of accesses (n), while the vertical axis indicates the download time. The download time for $n = 0$ is that of a conventional WWW server.

For page A, the download time for the first visit was far from reaching the specified transmission time threshold (20 seconds). Bandwidth estimation did not work well due to the rather large size of the HTML file which had to be highly compressed via the PPP connection. Another important point is that, in the case of 9.6 kbps, the download times greatly exceeded the specified transmission time threshold in both graphs because the bandwidth was too narrow, and the image reduction rate reached its lower limit (0.25). In the case of 10 Mbps, no quality adjustment was performed because it was not necessary. After the second page access, we were able to get the desired download times with the exception of the 9.6 kbps tests.

Note that the curves for 57.6 kbps oscillate in Figure 9(a), and the curves for 115.2 kbps oscillate in Figure 9(b). This oscillation was caused by bandwidth estimation error as follows. First, in both cases, the page download times in the case without adjusting image quality are near the specified transmission time. If the estimated bandwidth is judged to require quality adjustment to meet the specified transmission time constraint, the data sizes of the inline images are adjusted by degrading quality, and the transmission time becomes short. On the other hand, if the estimated bandwidth is judged not to require quality adjustment, the original images are transmitted, and the transmission time becomes long.

To refine the bandwidth estimation, a network-level bandwidth measurement mechanism should be integrated into the content delivery mechanism. We will discuss this point in Section 8.

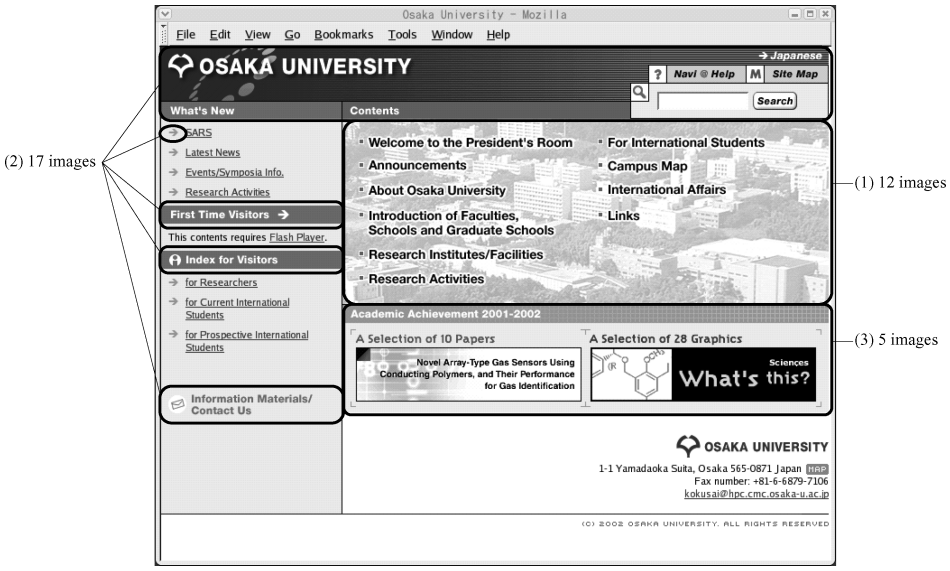


Fig. 10. The Web page used in the experiment.

5.2 Protocol Overhead of Transmission Order Control

If the transmission order description does not include parallel transmission(s), then there is no overhead for controlling the transmission order of inline objects. In this experiment, to show the protocol overhead of transmission order control when the transmission order specification includes parallel transmission(s), we compared the download times of normal operation without transmission order control and proposed behavior with transmission order control. Note that we used a minimum set of protocol headers in each method, and the transmission time threshold is not specified because we want to examine pure protocol overhead for parallel transmission of inline objects.

We performed two experiments.

(1) For measuring pure overhead of parallel transmission, we used a simple Web page that contains four photo images whose transmission order is specified as parallel. In order to examine how the data sizes of inline objects affect the overheads we used the following two sets of photo images:
   (1-a) The data sizes total about 88 KB (LARGE).
   (1-b) The data sizes total about 38 KB (SMALL).

(2) To measure the overhead when we access an actual Web page, we used the top page of our university, shown in Figure 10, as a typical Web page representing the top pages of Web sites. The data size of the HTML document is about 18 KB, which includes 38 images whose data sizes range from 67 bytes to 7,086 bytes. The total data size of the page is about 113 KB. Because this kind of Web page is considered a transit page, the images for links should be presented as quickly as possible. Therefore, we specified the transmission order for this page such that 12 images for links are

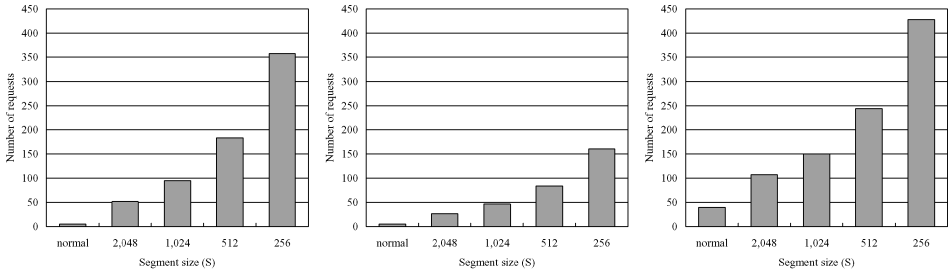


Fig. 11. Segment size vs. numbers of requests.

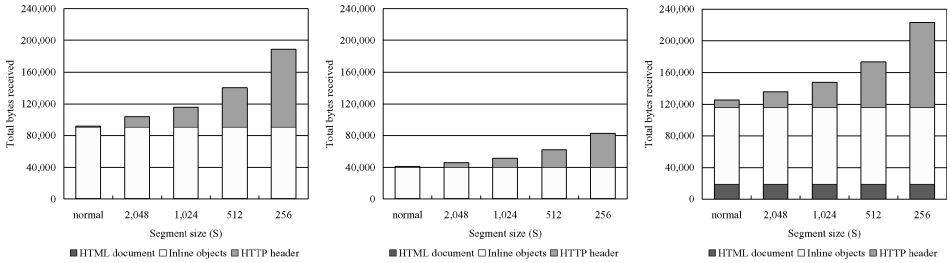


Fig. 12. Segment size vs. total bytes received.

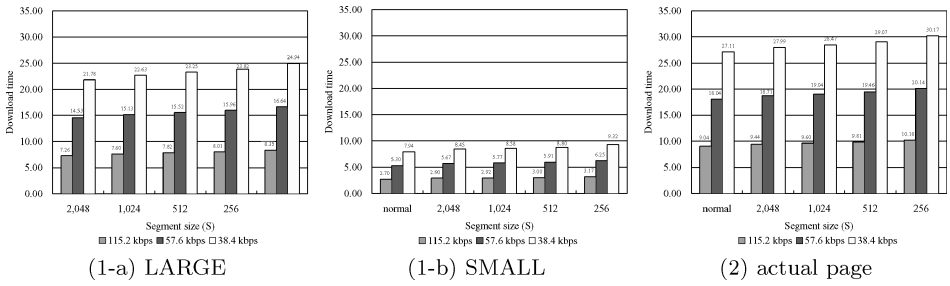


Fig. 13. Segment size vs. download time.

transmitted first in parallel (1), then 17 images laid at the top and the left of the page are transmitted in parallel (2), 5 images for links to special contents are transmitted in parallel (3), and lastly, the remaining 4 images are transmitted sequentially.

The experiments were performed for a PPP connection on an RS-232 serial link, while varying the serial port speed at 38.4, 57.6, and 115.2 kbps. A data compression option was used on the PPP connection. This option simulates narrow-band users using a modem because typical modems have data compression capabilities of about V.42bis. For parallel transmissions, we varied the segment size S to be retrieved by each range request.

Figures 11, 12, and 13 show the number of requests issued by the client, the total bytes received by the client, and the total download times, respectively.

From these results, we can see how the number of requests and the total bytes received increase as the segment size decreases. In experiment 1-a, the total bytes received increased about 26.2% when the segment size was 1,024 bytes, and increased about 105.2% when the segment size was 256 bytes. However, the download time did not increase much. This is due to the data compression on the PPP connection. The results of experiment 1-b show the same tendencies. These results indicate that the segment size is an important factor influencing the increase of the total number of requests and the download time. For example, if a client whose available bandwidth is 38.4 kbps wants to keep the increase of download time shorter than one second and the total data size to be transferred in parallel is large, the segment size should be set to 2,048 bytes or larger. On the other hand, if the total data size to be transferred in parallel is small, the segment size can be set to 1,024 bytes or 512 bytes. The results of experiment 2 exhibit the same tendencies as experiment 1, which show the feasibility of the transmission order control.

Note that if a client cannot enjoy data compression, the download time increases in proportional to the total data size to be transferred. Such a client should perform the parallel transmission only when the total data size in the parallel transmission is large, and the client should set the segment size to a value large enough to avoid much increase in the total number of requests.

5.3 Actual Impact of Transmission Order Control

Figure 14 shows the actual browser behaviors for experiment 2 of the previous section, where the connection speed was set to 38.4 kbps and the segment size was set to 1,024 bytes. The left side of the figure shows the normal behavior of the browser, while the right side of the figure shows the browser behavior when the client employs the transmission order control. As the figure shows, the inline images are usually presented from top to bottom and from left to right on the browser because the browser requests them in the order of `img` tag appearances in the HTML document. A normal transmission order such as in this Web page may not be effective. With transmission order control, the presentation of the images for links is almost complete as of 16 seconds which allows users to move to the page that s/he wants to visit.

Figure 15 shows the screenshots for another page, that is, the history page of our university. The transmission order was specified so that the images in the page's main content are transmitted first in parallel, and the remaining images are transmitted sequentially after that. Users who visit this page must be interested in the main content of the page, and the proposed method can present this content within 8 seconds. In this way, if the transmission order is specified appropriately, the proposed method can reduce user-perceived latency.

In this experiment, we showed a potential of transmission order control. Our conclusion is that by using our proposed framework, an appropriately-specified transmission order can reduce user-perceived latency. The trend of current Web page creation is the use of a three-pane design that has a left-side navigation area, a top navigation area, and a main content area. The Web pages used in this experiment are in line with such a design rule. For a Web

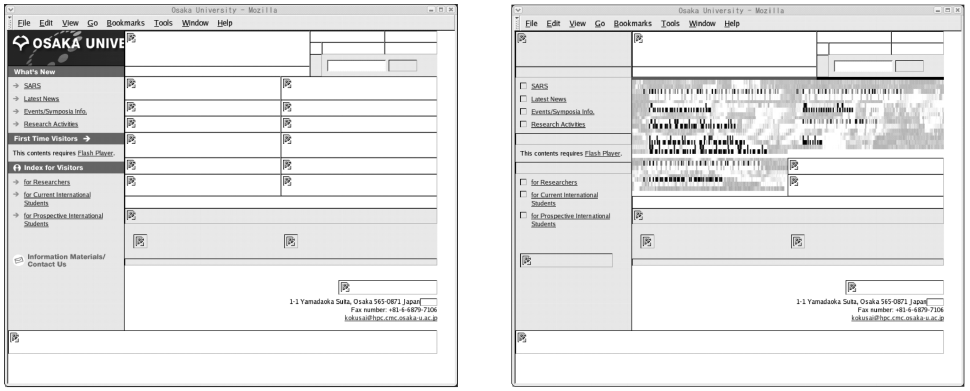


Fig. 14. The presentation of the top page. Left is normal operation, right is with transmission order control.

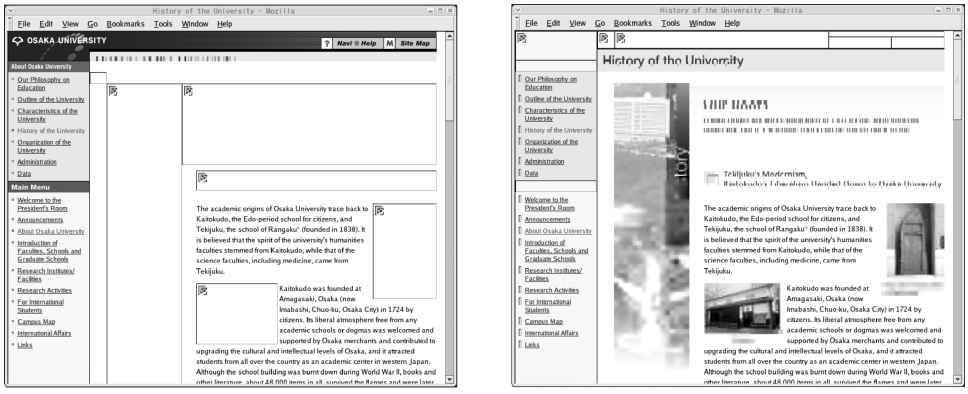


Fig. 15. The Presentation of the history page. Left is normal operation, Right is with transmission order control.

page that has the three-pane design, inline objects in the navigation areas are usually presented first, with inline objects in the main content area presented next. Using transmission order control, the browser can first present inline objects in the main content area which will reduces the user-perceived latency. However, inline objects in the navigation areas are likely to be cached because they are generally included in multiple Web pages of that site. In such cases, the reduction of user-perceived latency will be smaller.

Furthermore, the transmission order specified by the content author might not fit the user's demand, and then the user-perceived latency might increase. The transmission order control mechanism, however, is client-driven. Therefore, the client has the chance to change the request order of the inline objects regardless of the transmission order specified by the content author. Such a client-driven transmission order control is described in Section 6.2.

Finally, we note that the amount of reduced latency may change every time we download a page, depending on several factors such as the degree of network congestion and the server load. However, our proposed mechanism can guarantee that inline objects are transmitted in the specified order in any situation.

6. A-POLICY SPECIFICATION BY CONTENT AUTHORS AND END USERS

This section briefly describes two methods of specifying a-policies on Web pages, by content authors and by end users. Section 6.1 describes the development of an authoring tool equipped with an a-policy editor and a template-based a-policy generator which enables content authors to easily specify a-policies for their Web pages. In Section 6.2, since an appropriate set of a-policies should depend on the Web page viewer as well, we introduce preference profiles in order to reflect viewers' preferences for the kind of a-policies that should be created and then show an example of our adaptive content delivery.

6.1 Specification by Content Authors

Although the proposed framework achieves adaptive content delivery based on an a-policy, the work of describing a-policies for each Web page is labor intensive for content authors. To minimize this time-consuming work, we implemented a template-based a-policy generator. A template represents a user's desired a-policy at a semantic level, for example, all product images should be delivered before advertisement images. Without directly specifying a-policies, content authors can specify their demands at a semantic level using templates which are easily reusable and can be applied to other Web pages.

A template is described in XML format, and content authors can describe it using four kinds of elements: `first`, `last`, `transfer`, and `parallel`. These elements have a `role` attribute that indicates the target of the directive. The `transfer` element has either a `before` attribute or an `after` attribute that represents the relative transmission order of inline objects.

Figure 16 shows an example description of a transmission order template. This template means that 1) a background image should be transmitted last, 2) advertisement images should be transmitted in parallel with product images, 3) images of new products should be transmitted before images of other (not

```
<dto-template>
  <last role="background"/>
  <parallel role="ad or product"/>
  <transfer role="new and product" before="product"/>
  <parallel role="new and product"/>
  <first role="new and product"/>
</dto-template>
```

Fig. 16. Example description of transmission order template.

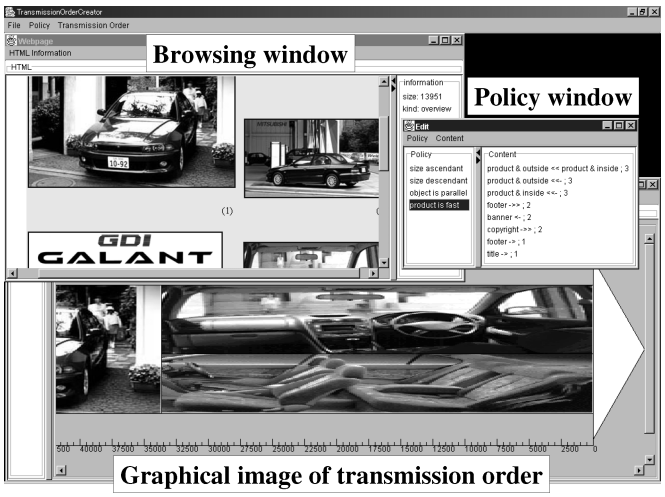


Fig. 17. Authoring tool for describing a-policies.

new) products, 4) images of new products should be transmitted in parallel, and 5) images of new products should be transmitted first.

The a-policy generator first loads one or more Web pages specified by content authors, and using predefined object metadata, identifies the semantic meaning of each inline object in the Web page. Next, the a-policy generator applies the template to the Web pages and generates HTML documents in which a set of a-policies are embedded.

Figure 17 shows an overview of the a-policy generator. In addition to a GUI for editing templates, it is equipped with functions to assist the content author's work such as a browsing window to view the metadata of inline objects, a graphic visualization of the transmission order, and a function to preview a Web page at an arbitrary bandwidth. Using this authoring tool, content authors will be able to handle a large number of Web pages in our content delivery framework.

6.2 Specification by End Users

6.2.1 *Preference Profiles.* To semantically represent a user's preferences and translate them into an a-policy, we introduce a preference profile grammar based on the RDF (Resource Description Framework) [W3C-RDF 2004] in which an adaptation policy is described. This policy includes the user's

```
<?xml version="1.0"?>
<rdf:RDF>
  <rdf:Description about="Default">
    <pp:AcceptableDownloadTimeSec>15</pp:AcceptableDownloadTimeSec>
    <pp:ContentQualityAdaptationPolicy>
      <pp:Priority role="new_info" value="1"/>
      <pp:Priority role="ad" value="-3"/>
    </pp:ContentQualityAdaptationPolicy>
    <pp:DeliveryOrderAdaptationPolicy>
      <pp:Priority role="new_info" value="2"/>
      <pp:Priority role="main_content" value="1"/>
      <pp:Priority role="ad" value="-2"/>
    </pp:DeliveryOrderAdaptationPolicy>
  </rdf:Description>
  <rdf:Description about="http://www.myfavorite.com/">
    <pp:AcceptableDownloadTimeSec>60</pp:AcceptableDownloadTimeSec>
  </rdf:Description>
</rdf:RDF>
```

Fig. 18. Example description of a preference profile.

acceptable download time threshold, content quality priority, and delivery order priority. Based on the user's preference profile, the content delivery mechanism dynamically prioritizes inline objects according to the user's specified content quality, and delivery order, and this provides the user with adapted inline objects. Adopting a standard framework like RDF makes it possible to share a preference profile among similar Web services provided at different Web sites.

Figure 18 shows an example description of a preference profile. We assume that each inline object has metadata describing itself which consists of multiple attribute-value pairs. For example, an attribute includes advertisement, animation, and decoration, while their values are boolean, yes or no. The preference profile also consists of multiple attribute-value pairs, where each value represents the importance of its corresponding attribute. By matching metadata to the preference profile, the content delivery mechanism generates a-policies such as those regarding quality prioritization and transmission order which provide an adaptive content delivery personalized for each user.

6.2.2 *Example of Adaptive Content Delivery.* We have extended the content delivery mechanism so that it generates a-policies based on a given preference profile and have also implemented a Web browser with a simple interface for editing profiles. Figure 19 illustrates an example of adapted content delivery.

For this example, we prepared a product lineup page for an online shopping site in which 13 images totaling 120 KB are embedded and then finally scaled down to 50.9 KB. The content is formatted in an XML document where the metadata for each object is described. The preference profile used in this experiment is the same as the one shown in Figure 18.

In our adapted content delivery system, the inline objects appear on the Web browser as follows. A high-quality image of a new model appears first (two seconds). After all new models are shown, images of current models appear at normal quality (six seconds). The rest of the product images then appear.

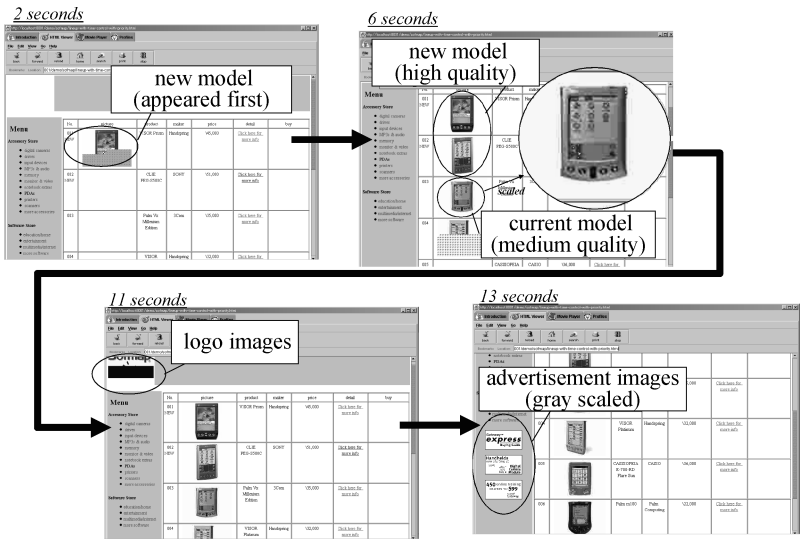


Fig. 19. Example of adaptive content delivery.

Next, logo images appear (11 seconds), and then advertisement images that have their quality degraded to a gray scale appear (13 seconds). Finally, icon images appear. The total delivery time is about 15 seconds, which is the user's acceptable download time configured in the preference profile. In the example of content delivery without adaptation, the time taken to complete the Web page is about 26 seconds.

In order for users to configure their own profiles, we have provided them with a simple interface. However, since the manual configuration might be labor intensive for some users, we are investigating an effective way to detect user preferences automatically. User preferences might be obtained by analyzing an access history of Web pages, enabling the delivery of an inline object with the desired quality and in the preferred order. Our content delivery adaptation may also benefit from the client clustering or characterization technique proposed in Krishnamurthy and Wills [2002] if there is no explicit preference specified by a client or there is no network characteristic of it available.

7. RELATED WORK

From the architectural point of view, there are two approaches to realizing Web content transcoding mechanisms: the proxy-based approach and the server-based approach.

Much work has been done on transcoding mechanisms based on the proxy-based approach [Brooks et al. 1995; Fox and Brewer 1996; Fox et al. 1998a, 1998b; Han et al. 1998; Shimada et al. 1997]. The proxy-based approach can be easily applied in real Internet infrastructure since there is no need to replace existing clients and servers. However, most of these studies give a proxy a transcoding function regardless of the intentions of content authors. Therefore, Web content may be altered without permission or recognition of its author.

Mogul [2000] tries to reflect content authors' intentions by a server-directed transcoding mechanism in which a transcoding proxy takes advantage of explicit directives defined by the origin server (e.g., minimal quality factor for an image) to maintain acceptable quality. Several methods have been reported to carry the directives to the transcoding proxy, including the introduction of a new HTTP header and the use of transcoding applets. These authors suggest that quality control be performed at the proxy placed at the edge of the Internet, which is often an access point for Internet users, in order to hide the bottleneck associated with the client side's low-speed link. Thus, only users who are connected to a proxy with a transcoding mechanism can enjoy the benefits of transcoding.

The server-based approach has also been studied, and several commercial products are now available such as IBM WebSphere Transcoding Publisher [IBM Corporation]. Krishnamurthy and Wills [2002] proposed a server-based adaptation mechanism in which clients are categorized into three classes (poor, normal, and rich) according to measured delays, and then the server takes an action such as the choice of content variants based on the classification. They use fixed criteria for client classification regardless of page content, for example, if a page transmission time is longer than 8 seconds, the client is categorized as poor. By contrast, our approach uses dynamic quality adjustment based on a transmission time threshold attached to each Web page. We believe that this is the right approach for serving appropriate Web content to a variety of users because content transmission speed from Web servers and end users is becoming increasingly diverse.

Hori et al. [2000] proposed an annotation-based transcoding approach. They used annotations as content adaptation hints for a transcoding proxy. Although they used the term transcoding proxy, their approach can be categorized as a server-based approach since their transcoding proxy can be regarded as a reverse proxy.

The transcoding system proposed in Smith et al. [1998] considers the conversion between different media types (e.g., image to text). The system evaluates the possible alternatives of certain content with respect to the fidelity and modality based on the priority specified by the users, and the best alternative is selected. In addition to defining a set of policies (a-policies) on how to adapt a Web page as in their frameworks, we have also explored a mechanism to define the delivery of adapted Web pages, namely the transmission order control mechanism, and the policy on the transmission order is integrated as one of the policies for effective Web browsing.

The server-based architecture has the advantage of making it easier for content authors to control the quality of service of their Web pages. In our system, for instance, the author can preview the quality of an adapted Web page by using the authoring tool we have developed and he can then revise the a-policy applied to the Web page. Moreover, since the quality is controlled at the content server, this approach has no problem with copyright infringement, that is, content authors have full control of content quality.

Another approach to controlling the quality of service in Web content delivery exists [Chandra et al. 2000; Abdelzaher and Bhatti 1999; Eggert and

Heidemann 1999; Pandey et al. 1998; Bhatti and Friedrich 1999; Almeida et al. 1998]. In these works, the quality of service for Web pages is controlled to address the performance problems of Web servers caused by overloading. In Eggert and Heidemann [1999], Pandey et al. [1998], Bhatti and Friedrich [1999], and Almeida et al. [1998], for example, user requests are prioritized and scheduled in order to guarantee that high priority requests are processed even in a heavily-loaded situation. In Chandra et al. [2000], and Abdelzaher and Bhatti [1999], to provide finer-grained differentiated services, the compression rates of JPEG images are changed. However, none of these works mention how the content authors or system administrators should specify the quality of service constraints except for Pandey et al. [1998], where a special syntax is introduced to define how much server resource should be allocated for a particular Web page to be served. Consequently, a more flexible framework for quality control is necessary.

After the quality of a Web page is adjusted, delivering the adapted Web page efficiently and effectively becomes important. The globally progressive Web delivery proposed in Gilbert and Brodersen [1999] is intended to improve Web access latency by an interaction between a Web browser and a customized Web proxy server. The proxy transforms all of the embedded images in a Web page to Java applets so that it can control the delivery of embedded images. All of these images are concurrently delivered in such a way that they are progressively shown on a Web browser. This proxy-based approach, again, has the advantage in that there is no need to modify current Web browsers and servers, but this system provides no mechanism allowing content authors to explicitly specify the transmission order of embedded objects.

Other types of efficient Web content delivery are presented in Wills et al. [2001], where embedded objects in a Web page are bundled and delivered from a server to a client. Through a measurement study, it was shown that this bundling can reduce the time for a client to retrieve and render a Web page. In their scheme, however, embedded objects can be bundled only in a sequential order, not in parallel. Parallelized delivery is easily enabled by the transmission serialization in our proposed mechanism and has the potential to reduce user-perceived latency as shown in Section 5.3. As in the study in Mogul et al. [1997], the authors also suggest a way to reduce retrieval time of a Web page through the use of data compression. However, the decompression of a set of embedded objects requires the entire data to be downloaded, and thus nothing can be presented on the user's Web browser during the downloading of the compressed data which may increase user-perceived latency.

8. CONCLUSIONS AND FUTURE WORK

In this article, we described an adaptive content delivery mechanism for the WWW that optimizes a single Web page and adapts it to a heterogeneous group of Internet users. This adaptation is performed based on a-policies that describe transmission time threshold, quality prioritization, format conversion conditions, and transmission order. Through adaptive content delivery, a user

is presented with the highest quality Web page possible in terms of presentation and speed. We also described useful methods to specify a-policies both by content authors and by end users.

We consider the proposed specification language sufficient to describe a-policies in most cases. However, more fine-grained specification may help content authors and end users to control adaptation of Web pages. For example, although we allow image prioritization by abstract terms (high and low), more precise specification such as image prioritization by more fine-grained values and specification of a lower limit of reduction rate may help us to maintain image semantics.

Furthermore, some implementation issues have become apparent by the experiments. Among them is the need for a more precise bandwidth estimation to perform stable delivery of Web pages. In Krishnamurthy and Wills [2002], some techniques for estimating the round trip time and bandwidth are presented that can be used for Web adaptation. In addition, this issue has been extensively studied by network researchers (e.g., Jain and Dovrolis [2002]), and our content delivery mechanism would benefit from the adoption of such a network-level bandwidth measurement technique.

Another issue to be explored is adaptive determination of the segment size for parallel transmission based on the available bandwidth, the data size to be transferred in parallel, and allowable download time increase. It should also be adapted according to whether data compression is used between the server and the client or not.

In addition, in order to evaluate our proposed framework in more detail and to issue guidelines regarding the use of our proposed mechanisms, we will experiment in an actual publishing environment. In the experiment, for example, we will focus on an online shopping site and discuss how the transmission order should be controlled to reduce user-perceived latency and to avoid premature transmission termination by users due to long download times.

Finally, we plan on pursuing dynamic content prioritization based on both content author specifications and user preferences. In such a system, there might arise the need to develop a coordination mechanism to satisfy both demands. Furthermore, from a technological point of view, we realize that we need to develop and integrate various modules into our system. For example, we need to implement a profiling module to automatically extract user preferences such as acceptable download time and priority for inline objects. A bandwidth-monitoring module combined with a TCP-monitoring function should be integrated to precisely measure effective bandwidth. The msd should be extended to handle not only images but also audio and video. In addition, other modules need to be considered for request scheduling and load balancing and then integrated into our system.

ACKNOWLEDGMENTS

We would like to thank two anonymous reviewers and the editor Balachander Krishnamurthy for their comments that led to significant improvements in this article.

REFERENCES

ABDELZAHER, T. F. AND BHATTI, N. 1999. Web content adaptation to improve server overload behavior. In *Proceedings of the 8th International World Wide Web Conference*. Toronto, Canada. 1563–1577.

ALMEIDA, J., DABU, M., MANIKUTTY, A., AND CAO, P. 1998. Providing differentiated levels of service in web content hosting. In *Proceedings of ACM SIGMETRICS Workshop on Internet Server Performance*. Madison, Wisconsin, WI.

BHATTI, N. AND FRIEDRICH, R. 1999. Web server support for tiered services. *IEEE Network 13*, 5, 64–71.

BROOKS, C., MAZER, M. S., MEEKS, S., AND MILLER, J. 1995. Application-specific proxy servers as HTTP stream transducers. In *Proceedings of the 4th International World Wide Web Conference* Boston, MA. *WWW J 1*, 1, 539–548.

CHANDRA, S., ELLIS, C. S., AND VAHDAT, A. 2000. Application-level differentiated multimedia web services using quality aware transcoding. *IEEE J. Select. Areas Comm. 18*, 12, 2544–2565.

EGGERT, L. AND HEIDEMANN, J. S. 1999. Application-level differentiated services for web servers. *WWW 2*, 3, 133–142.

FOX, A. AND BREWER, E. A. 1996. Reducing WWW latency and bandwidth requirements by real-time distillation. In *Proceedings of the 5th International World Wide Web Conference*. Paris, France. 1445–1456.

FOX, A., GOLDBERG, I., GRIBBLE, S. D., LEE, D. C., POLITO, A., AND BREWER, E. A. 1998a. Experience with top gun wingman, a proxy-based graphical Web browser for the 3Com PalmPilot. In *Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*. The Lake District, England. 407–424.

FOX, A., GRIBBLE, S. D., CHAWATHE, Y., AND BREWER, E. A. 1998b. Adapting to network and client variation using active proxies: Lessons and perspectives. *IEEE Pers. Comm. 5*, 4, 10–19.

FRYSTYK, H., 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 ACM SIGCOMM'97*. Cannes, France. 155–166.

GILBERT, J. M. AND BRODERSEN, R. W. 1999. Globally progressive interactive web delivery. In *Proceedings of the Conference on Computer Communications (IEEE INFOCOM)*. New York, NY. 1291–1299.

HAN, R., BHAGWAT, P., LAMAIRE, R., MUMMERT, T., PERRET, V., AND RUBAS, J. 1998. Dynamic adaptation in an image transcoding proxy for mobile web browsing. *IEEE Pers. Comm. 5*, 4, 8–17.

HORI, M., KONDOH, G., ONO, K., HIROSE, S., AND SINGHAL, S. 2000. Annotation-based Web content transcoding. In *Proceedings of the 9th International World Wide Web Conference*. Amsterdam, Netherlands. 197–211.

IBM CORPORATION. WebSphere Transcoding Publisher. Available at `http://www.ibm.com/software/pervasive/transcoding_publisher/`.

JAIN, M. AND DOVROLIS, C. 2002. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. In *Proceedings of ACM SIGCOMM'02*. Pittsburg, PA. 295–308.

KRISHNAMURTHY, B. AND WILLS, C. E. 2002. Improving Web experience by client characterization driven server adaptation. In *Proceedings of the 11th International World Wide Web Conference*. Honolulu, Hawaii. 305–316.

MOGUL, J. C. 2000. Server-directed transcoding. In *Proceedings of the 5th International Web Caching and Content Delivery Workshop*. Lisbon, Portugal.

MOGUL, J. C., DOUGLIS, F., FELDMANN, A., AND KRISHNAMURTHY, B. 1997. Potential benefits of delta encoding and data compression for HTTP. In *Proceedings of ACM SIGCOMM'97*. Cannes, France. 181–194.

PANDEY, R., BARNES, J. F., AND OLLSSON, R. 1998. Supporting quality of service in HTTP servers. In *Proceedings of the 17th ACM Symposium on Principles of Distributed Computing*. Puerto Vallarta, Mexico. 247–256.

RFC2616. 1999. Hypertext transfer protocol—HTTP/1.1. Available at `http://www.ietf.org/rfc/rfc2616.txt`.

SHIMADA, T., IWAMI, N., TOMOKANE, T., HAYASHI, M., AND KUWAHARA, Y. 1997. Interactive scaling control mechanism for World Wide Web systems. *Comput. Netw. ISDN Syst. 29*, 8–13, 1467–1477.

SMITH, J. R., MOHAN, R., AND LI, C.-S. 1998. Transcoding Internet content for heterogeneous client devices. In *Proceedings of the International Symposium on Circuits and Systems*. Monterey, CA. Vol. 3. 599–602.

W3C-JIGSAW. Jigsaw—W3C's Server. Available at `http://www.w3.org/Jigsaw/`.

W3C-RDF. 2004. RDF primer. W3C Recommendation. Available at `http://www.w3.org /TR/rdf-primer/`.

WILLS, C. E., MIKHAILOV, M., AND SHANG, H. 2001. N for the price of 1: bundling Web objects for more efficient content delivery. In *Proceedings of the 10th World Wide Web Conference*. Hong Kong. 257–265.

Received August 2002; revised February 2003, May 2004; accepted July 2004