An Active Transcoding Proxy to Support Mobile Web Access*

Harini Bharadvaj, Anupam Joshi and Sansanee Auephanwiriyakul
Department of Computer Engineering & Computer Science,
University of Missouri-Columbia,
Columbia, MO 65211 USA
E-mails: {harini,sansanee}@meru.cecs.missouri.edu, joshi@cecs.missouri.edu

Abstract

In this paper, we present a proxy based system (MOWSER) to support web browsing from mobile clients over wireless networks. Mowser is a proxy agent between the mobile host and the web server, which performs active transcoding of data on both upstream and downstream traffic to present web information to the mobile user according to the QoS parameters set by the user. Active transcoding is defined as modifying the HTTP stream in situ, and it is entirely transparent to the user. Further, our system does not pose any additional requirements on the mobile user. This is an improvement over other proxy based systems, which only transcode images on the downstream and are mostly not configurable. While developed for mobile users, such a system can actually be useful in any low bandwidth scenario.

1. Introduction

With an ever increasing amount of information "out there" on the World Wide Web, and mobility becoming the need of the hour, users want to have information at their fingertips wherever they may be. Recent developments in mobile computing and web technologies have resulted in an increase in the number of "road warriors", people who use mobile computers to access their business information repositories, often through the web. These users need speedy access to that data. One cannot expect them to wait for hours to download a web page containing a lot of multimedia data over a wireless network, most of which cannot be handled by the small and "easy-to-carry" laptop/palmtop computer anyway. Most mobile computers have very limited cpu, memory & disk resources. They communicate over wireless links which are characterized by lower bandwidths, higher error rates, and more frequent disconnections. Moreover, in a mobile environment, changes to the network bandwidth and resources are very common. These are only some of the challenges of mobile computing, more of which are discussed in [4]. In the context of web browsing, web servers do not consider the network connection between them and the client. They also have no concern for the hardware capabilities of the client. They just return the document asked for assuming that the client is capable of properly receiving and displaying the data. People are commonly creating web pages rich in multimedia data, pages with several images and videos have become very common. Presenting all this multimedia-rich data over wireless networks to the mobile user is a major challenge. The mobile computer either does not have the appropriate hardware or adequate bandwidth (or both) to handle the content, and the web servers also cannot be modified to suit the mobile user. Therefore, the only way of bridging the gap between the highly resource-rich web servey at one end and the highly resource-poor mobile client at the other, is by introducing a system in between which modifies the contents on the web and present it in the most appropriate form to the mobile user. Hence, the need for middleware, which adapts to the mobile environment, gets the best of what is available on the web and does not pose any additional requirements on the mobile client.

We have devised such a middleware based solution which allows the user of a mobile computer to control the way in which the data from the web is retrieved, dynamically transform the data in a way that is transparent to the user, without requiring the mobile computer to do any additional work. We achieve this by introducing a proxy agent between the mobile client and the web server. Our proxy, Mowser, lets the user specify and control the viewing preferences and hardware capabilities of the client, transforms the data to and from the web server, without requiring any change on the client. We talk about related work in section 2 and describe the software architecture of our system in section 3. In section 4, we describe the various methods of transforming data that we have used. The results of our experiments are presented in section 5 and we discuss it in section 6.

2. Related Work

The Client-Proxy-Server model has begun to feature in many mobile applications to overcome the challenges faced in the mobile computing scenario. However, only some of them actually transform the stream between the client and the server.

In the GloMop model described in [5],[11] the proxy performs "distillation" of the document received from the server before sending it to the client. Distillation is defined here as highly lossy, real-time, datatype-specific compression that preserves most of the semantic content of a document. The approach of transcoding image files is similar to our approach first outlined in [9], but video files are handled

*This work was supported in part by an IBM Faculty Development Award to A. Joshi

differently. They perform real-time transcoding of motion JPEG to sub-sampled H.261, while our approach, described later in this paper, represents a video stream using representative frames. GloMop also allows refinement of selected portions of the document.

The Mowgli model [14] consists of two mediators, the Mowgli Agent and the Mowgli Proxy located on the mobile host and the mobile-connection host respectively. They use the Mowgli HTTP protocol to communicate with each other, which reduces the number of round-trips between client and server. A specialized transport service, the Mowgli Data Channel Service is used for reliable communication between the mobile-connection host and the mobile host. Mowgli WWW reduces the data transfer over the wireless link in three ways: data compression, caching, and intelligent filtering. It only performs GIF to JPEG conversion, and large embedded images are not transferred at all to the mobile node.

Zenel proposes an architecture in [19] in which the proxy is made up of three components: High Level Proxy, Low Level Proxy and Event Manager. The High Level Proxy allows filters for application layer protocols to be downloaded dynamically from mobile host applications. The Low Level Proxy is used to create and install filters for the transport and network layers. A control interface is provided for the filters running within these proxies by the Event Manager.

HTTP transducers called OreOs (defined in [3]) are specialized processing modules having four classes of functionality: filtering HTTP requests and responses, characterizing sets of messages, transforming message contents and additional processing indicated by the messages. The "stream model" approach of signal processing is used to perform complicated processing.

In the work of Sathyanarayanan et. al, a module called Cellophane [17] on the client transforms HTTP requests from Netscape into file operations on Odyssey [16] web objects and makes use of Odyssey API to select fidelity levels for images which are forwarded to a distillation server. The distillation server distills the images to the requested fidelity before passing them back to the client. However, this approach is specific to the Odyssey file system and requires a modified version of the Net BSD kernel. This also requires the addition of a module on the client.

Intel's Quick Web Technology [7] which sits on ISP servers compresses images by selectively dropping bits or pixels out of an image using lossy compression techniques, thereby speeding up the download of graphically- rich web pages. It also caches data to overcome the problem of bandwidth bulge. This can be used only when the access to the Internet is through ISP, and the user requires a Java-enabled browser to have control over the compression.

IBM's Web Express [6] consists of two components: ARTour (Advanced Radio Communications on Tour) Gateway and ARTour Client. The Gateway provides secure, compressed data across the selected network with authentication. It can automatically retrieve Web requests in the background while mobile users are performing other tasks.

In our system, the proxy performs active transcoding of HTTP requests from the client while sending it to the server, according to the preferences set by the mobile host, so that the document in the most suitable format is retrieved. It also processes the received HTTP data before sending it to the mobile host if necessary.

3. Software Architecture

Mowser is a proxy HTTP server agent (written in Perl) [18] which allows a mobile user to specify his or her viewing preferences, based on the network connection and available resources, and performs active transcoding of HTTP streams accordingly. The software architecture that we propose introduces a proxy server on each Mobile Support Station (MSS) to the basic MSS-MH (Mobile Host) model which accepts and stores the preferences for each of its mobile hosts, acts as the server to the mobile host, and as a client to the WWW server. No modifications to the web client on the MH is required. So any WWW browser that can handle forms and has the provision of a proxy can be used. No additional software is required on the mobile host. Also, setting and updating preferences is done by just filling up a CGI form on a URL at the web site maintained by the proxy server.

In the initial versions of Mowser [9],[10], the proxy dealt with getting the viewing preferences for a MH from the user and storing it according to its IP address. The current version also stores the accept headers that will be most suitable for the MH based on the preferences set by the user. The viewing preferences stored for each MH include a starting point, color capability, video resolution, sound capability, maximum allowed size for text, image, video, audio files and files of unknown type, and the size reduction technique for image files. Not all of these variables are presently used by the proxy. The preferences can be updated by the user of the MH whenever there is a change in the network connection or available resources. We are using an Apache HTTP server to store the preferences, and CGI scripts written in Tcl and Perl are used to update and save the preferences.

Once the MH sets the Proxy server as its proxy, all communication between the MH and the WWW servers is directed through this proxy. When the proxy receives a request from the MH, it looks up the preferences stored with the IP address of the MH and processes the request accordingly. Default preferences are used if no preferences had been specified. The proxy processes requests to set preferences and the GET requests. All other requests are forwarded to the target WWW server. In the next section, we detail how the proxy performs active transcoding of HTTP streams.

4. Active Transcoding

Modifying the HTTP stream and changing its content in situ is called active transcoding. This is done dynamically without any user intervention. For example, if an image file does not meet the size or color specifications, it is reduced before being sent to the MH (described in section 4.2). Similarly, sound files will not be sent to a MH with no sound capabilities, and so on.

Traditionally, transcoding is a unidirectional process [9], [5],[6],[7]. In other words, the request from the client is passed as is to the target server, while the return stream's multimedia content is altered. In our work, we alter the request as well, so as to take advantage of some net-friendly features of HTTP/1.1.

After setting preferences and making our MOWSER as its proxy, the user can browse the web as s/he would with any web client. On receiving a request from the MH, the proxy fetches the preferences set by the MH and serves the MH with files in the most suitable format. Default preferences are used if no values had been set by the MH. We process HTTP GET requests received from the MH before sending it to the WWW server, and modify image and video files received from the WWW server before transmitting it to the MH if necessary. To support MHs with very limited resources and hardware capabilities like PDAs, we even parse the HTML stream to remove the active content and any tags that the MH cannot handle. The user may even choose to block any HTML file greater than a given size. With transcoding being done at two steps independently as shown in Figure 1, we are making sure that we match the preferences of the MH, while using the wired bandwidth in the most efficient manner.

4.1. Transcoding of HTTP Requests

HTTP/1.1 introduces the concept of content negotiation. The basic idea is that a WWW server may have several different representations of a resource. For example, it may store a document as postscript or word or HTML, etc. The server can automatically choose the file to send if the client sends the preferred representations as part of each request. Most servers (e.g., apache), even though not fully 1.1 compliant, already support content negotiation and will store files in several formats and in several variations of a format. We use this idea to get the file in the most appropriate format for the MH. For example, an image file may be made available in varying resolutions by the content provider on the server. We request the server to send the image file which has the resolution appropriate to the present QoS and client parameters, by including the preference in the request. This requires that the variants of a file have different mime types. For example, in our experiments we have used image/x-sgif to denote an image file with very low resolution, image/x-mgif to denote one with medium resolution and image/x-lgif to denote one with large resolution. We have also introduced video/x-rmpg to denote representative frames of video files (discussed in section 4.3).

Any HTTP GET request received from the MH is munged to an HTTP 1.1 request and the complete URI is included in the request line. The Accept headers stored for the MH are then appended to the outgoing stream to request for the file in the format most suited for the MH. A Host header is added to complete the HTTP 1.1 request. The server performs content negotiation and sends the file which closely meets the format specified in the request. Thus, the process is transparent to the user, and works even if the request comes from an HTTP/1.0 compliant browser, like most present commercial systems.

For example, for a MH host on a low bandwidth line, the proxy may append the following Accept headers to the request after making it a HTTP 1.1 request:

```
Accept: image/x-sgif, video/x-rmpg
```

For a MH like the PalmPilot, which can handle only text and images, the proxy greatly reduces the data transfer by selectively GETing the files. That is, when the proxy receives a GET request from a PDA, it sends a HEAD request to the WWW server to get information about the content type of the file, and then GETs the file only if the PDA can handle it. For example, the proxy does not request for audio, video and application files for a PDA. Since a page has the URLs of additional files to be fetched embedded in it, we could prevent the client on the MH from generating the additional GET requests and design the proxy to decide whether to GET the file or not by just looking at the extension of the file name in the embedded URLs. However, the content-type of the file is a better indicator of the format of the file than the extension in the filename, though getting this information requires an additional HEAD request to be sent.

4.2. Transcoding Image Files

When the proxy finds an image tag in the HTTP stream received from the server, it reads the URI of the image file to be fetched and first sends a HEAD request to the server. It checks the content-type and content-length information received from the server. If the content-length is small enough to be handled on the MH, the image file is sent to the MH unmodified. But if the image is larger than what can be handled by the MH, it is reduced in size or color as requested by the MH. The image files are scaled down in size, or the number of colors is reduced, or both without sacrificing semantics. On an image map for instance, size is not changed, only colors are, to preserve the semantics. The content-type information is used to decide the transformations that the image file has to go through. We convert all images to be reduced to portable pixmap format for processing and then convert them back to gif format for displaying. Then the original URL in the image tag is replaced with the URL of the modified image stored locally by the proxy and sent to the MH. This makes the MH GET the modified image file from the proxy. To display images on PDAs, the proxy might have to reduce images to 2-bit gray scale and thumb size.

4.3. Transcoding Video Data

Unlike image data where transcoding steps are obvious, video data represents a great challenge. Simple subsampling, as proposed in [5], is still not adequate as some clients may not have enough computational resources to do software decoding of MPEG or H.261. We use the structure inherent in video streams to do the transcoding. The structure of video is a hierarchy of the movie or episode. This hierarchy is segments, scenes, and shots. Each segment consists of sequence of scenes, each scene consists of several shots, and each shot is composed of several frames which have similar visual properties. Thus one of these frames can be selected as a Representative frame (Rframe) for the shot.

We present the video to the user by the representative frames which are picked from each shot. Using techniques, we have developed [1],[8] to support content based access to networked video databases. We have used several fuzzy clustering algorithms, such as fuzzy c-mean, hard c-mean, fuzzy c-median, hard c-median and possibilistic c-mean [2],[13],[12]. We use luminance and chrominance features, and 1-norm and 2-norm distance measures [1], [8] in order to group the frames which have similar properties together. Each group is classified as one shot. We pick the frame that is closest to center of each group to be Rframe. We do not explicitly use any scene change detection algorithms. The fuzzy techniques are used since frames can belong to the clusters to different degrees (membership values). Traditional scene change algorithms, which insist on a frame belonging to only one group, break down when confronted with gradual scene changes typically found in videos. While Rframes can be computed dynamically by the proxy, we feel that from a computational perspective, this should be done at the server side. In fact, it can be argued [1],[8] that these will typically be available at the server side already to support querying and browsing of the video database.

4.4. Transcoding Active Content

For mobile hosts with limited memory and computational resources, the user can decide not to receive any Java applets, JavaScripts, VBScripts, etc. When such a preference is set, the document to be transmitted is parsed and all the active content is eliminated before sending it to the MH. This is specifically suitable for the PDAs which have very small disk space and low speed CPUs. Also, given restrictions on the memory footprints of the applications that can reside on such machines, it is not clear that browsers will be able to support the virtual machines needed for active content languages such as Java.

Often though, Java (and JavaScript) are used to provide functionality that can be duplicated using CGI callbacks or server parsed HTML. An interesting option that we wish to pursue here is to see if we could have the forms capable client request for a CGI version instead of the JavaScript from the server. In other words, replace active content of a page with equivalent dynamic content. This feature will be supported via content negotiation. Like all content negotiation, it assumes that the server provides alternative versions (CGI based vs Java based) of a particular URL.

4.5. Transcoding HTML

We can reduce the computation on the MH by parsing HTML tags on the proxy itself, rather that on the MH. We can eliminate all the tags that the MH does not support, and references to any file that the MH is not capable of handling. For example, we can eliminate the italics tag, cascade style sheets, etc. for a PDA such as the PalmPilot. For such severely resource constrained MHs, the set of tags that it can handle may be so small, that it is advantageous to strip of all unwanted tags at the proxy, and encode the remaining tags using a few bits.

By choosing the specific options, the user can use any or all of these transcoding methods depending on the limitations of the client or network connection, and can change it when resources change. This makes our proxy very adaptable to serve the varying needs of the user. For example, a user on a laptop may want to only limit the size of video and audio files when s/he is connected via a slow telephone modem, and remove this restriction when connected through the ethernet. A user on a PDA, on the other hand, will want to filter out everything except text and small images.

5. Experimental results

Our proxy server is a modified version of a HTTP server written in Perl. We are using an Apache server to store the preferences and to act as the WWW server capable of content-negotiation. We stored multiple formats of some files and requested them with different preference settings.

An example of transcoding due to content negotiation:

We set the following preferences for one computer (A desktop) Maximum Image file size = 20K Maximum Video file size = 500K

The accept headers added to its request were: Accept: image/x-lgif, image/gif;q=0.6, video/mpeg

We requested for the page http:// bochi. cecs. missouri. edu: 9021/ demo.html

Figure 2 shows the response received.

We set the following preferences for another computer (A laptop) Maximum Image file size = 6K Maximum Video file size = 25K

The accept headers added to its request were: Accept: image/x-sgif, image/gif;q=0.6, video/x-rmpg, video/mpeg;q=0.6

We requested for the same page http:// bochi. cecs. missouri. edu: 9021/ demo.html

Figure 3 shows the response received.

An example of transcoding of images received from the server:

For the same two computers, (same preferences set as above) we requested the page http://www.missouri.edu/˜csacm The image file existed only in gif format on the server. The image in the document is small enough for the desktop and hence passed through without any reduction as seen in Figure 4. But the image is large for the laptop (larger than 6K). Therefore, the proxy reduced the resolution of the image as seen in Figure 5.

To see some video files and their representative frames, please visit http://meru.cecs.missouri.edu/˜sansanee/mpeg

We kept different settings on two computers and accessed the same web page to see the difference. For the first computer we set a large value for the maximum size of image and video files allowed, and sent accept headers to allow large gif files and mpeg movie files. Therefore, we received large image files and the entire movie file. For the second computer, we limited the size of image and video files, and sent accept headers requesting for small images and representative frames of mpeg files. Hence, we received smaller versions of the images and only the representative frames for the movie file. Our request did not specify the extension of the file name, and the file was available in multiple formats. If the size of the received image files is larger than the maximum size specified, the proxy scales it down either by size or by color as set by the user.

Clearly, the results are best understood by experiencing the proxy based model. We have made the proxy available on the web, it may be accessed at the URL http://nirvana.cecs.missouri.edu:8001. A major drawback of the present implementation is its overhead. SInce this is a demonstation prototype, it has been mostly in PERL, and is thus may slow to execute.

6. Discussion

In this paper, we have presented a proxy based system (MOWSER) to support web browsing from mobile platforms. It follows the client-proxy-server model which is the basis of most mobile applications and uses the proxy to provide active transcoding. Proxies are mostly used for forwarding data between the mobile client and the stationary server. The idea of using transcoding at the proxy to support mobility is not new per se. Many proxy based systems [5],[14],[19] have been developed to provide web access to mobile users. However, they typically transcode the image data received from the WWW server before sending it to the mobile client, and are often not configurable. In Mowser, we extend the notion of transcoding to both the upstream and the downstream traffic. More specifically, the upstream request is munged into a HTTP/1.1 request, to make use of the content negotiation feature, and Accept headers are appended, to request for the document in the format most appropriate for the QoS parameters set by the mobile user. On the downstream, in addition to transcoding of images, we also provide the options of removing the active content and transcoding HTML. The mobile user can select any or all of these options. This is to support the changing requirements of a wide variety of mobile hosts ranging from a powerful notebook with 233MHz CPU, 2GB RAM, which may require only image and video data transcoding, to a PDA with 64KB dynamic RAM, which requires all possible transcoding and filtering of data.

In ongoing work, we are extending the proxy to effectively use all the preferences set by the user to limit or transform the data before serving the MH. Further, our proxy adds a performance overhead due to two reasons. First, it is written in Perl and uses netpbm for the processing of image files. The speed could be increased by writing optimized C code and image conversion routines. Second, messages go all the way up to the application layer in the proxy even if data just needs to be written from one socket to another. Research is going on in the IBM Watson Labs to avoid this by using TCP Splice [15] which allows data to flow through without going to the application layer in the proxy if necessary and such techniques could eventually be integrated into our system. The overhead is justifiable because we are trading proxy side CPU cycles for the more expensive client side CPU cycles and network bandwidth. In experimental situations, it has been observed that extra time taken by the proxy is still less than the time needed to send untransformed data on the wireless network.

References

[1] S. Auephanwiriyakul, A. Joshi and R. Krishnapuram, "Fuzzy Shot Clustering to Support Networked Video Databases," *IEEE FUZZ-IEEE 98/WCCI98* May 1998.

[2] J. C. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algorithms," *Plenum Press* New York, 1981.

[3] C. Brooks, M. S. Mazer, S. Meeks and J. Miller, "Application-Specific Proxy Servers as HTTP Stream Transducers," *Proc. WWW-4, Boston, http://www.w3.org/pub/Conferences/WWW4/Papers/56* May 1996.

[4] G. Forman and J. Zahorjan, "The Challenges of Mobile Computing," *IEEE Computer, 27:38-47* April 1994.

[5] A. Fox and E. A. Brewer, "Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillations," *Fifth International World Wide Web Conference* May 1996.

[6] IBM Corporation, "Ringing in Wireless Services: Web Access Without Wires," *http://www.ibm.com/ Stories/ 1997/08/ wireless4.html.*

[7] Intel Corporation, "Intel Quick Web Technology: White Paper," *http://www.intel.com/quickweb/white.htm.*

[8] A. Joshi, S. Auephanwiriyakul and R. Krishnapuram, "On Fuzzy Clustering and Content Based Access to Networked Video Databases," *8th IEEE Workshop on Research Issue in Data Engineering* 1998.

[9] A. Joshi, R. Weerasinghe, S. P. McDermott, B. K. Tan, G. Benhardt and S. Weerawarna, "Mowser: Mobile Platforms and Web Browsers," *Bulletin of the IEEE Technical Committee on Operating Systems and Application Environments* Vol 8, no. 1, 1996.

[10] A. Joshi, S. Weerawarna and E. N. Houstis, "On Disconnected Browsing of Distributed Information," *Proceedings of the seventh International workshop on RIDE* pp 101-107, IEEE Press, 1997.

[11] R. H. Katz, E. A. Brewer, E. Amir, H. Balakrishnan, A. Fox, S. Gribble, T. Hodes, D. Jiang, G. T. Nguyen, V. Padmanabhan, M. Stemm, "The Bay Area Research Wireless Access Network (BARWAN)," *Proceedings Spring COMPCON Conference* 1996.

[12] P. R. Kersten, "Fuzzy Order Statistics and Its Application to Fuzzy Clustering," *Noval Air Warfare Center Weapons division Report* August 1995.

[13] R. Krishnapuram and J. M. Keller,"A Possibility Approach to Clustering," *IEEE Transactions of Fuzzy Systems* vol. 1 No. 2, pp 98-110, May 1993.

[14] M. Liljeberg, H. Helin, M. Kojo, and K. Raatikainen, "Enhanced Services for World-Wide Web in Mobile WAN Environment," *Report C-1996-28* April 1996.

[15] D. Maltz and P. Bhagwat, "MSOCKS: An Architecture for Transport Layer Mobility," *IEEE Infocom 98, San Francisco* pp 1037-1045, April 1998.

[16] B. D. Noble, M. Price, M. Sathyanarayanan, "A Programming Interface for Application-Aware Adaptation in Mobile Computing," *Proceedings of the Second USENIX Symposium on Mobile & Location-Independent Computing* Ann Arbor, MI, April 1995.

[17] B. D. Noble, M. Sathyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, K. R. Walker, "Agile Application-Aware Adaptation for Mobility" *Proceedings of the 16th ACM Symposium on Operating System Principles*

[18] George Vanecek Jr., "Personal Communication," 1995

[19] B. Zenel. "A Proxy Based Filtering Mechanism for The Mobile Environment," *Thesis Proposal, Department of Computer Science, Columbia University.*

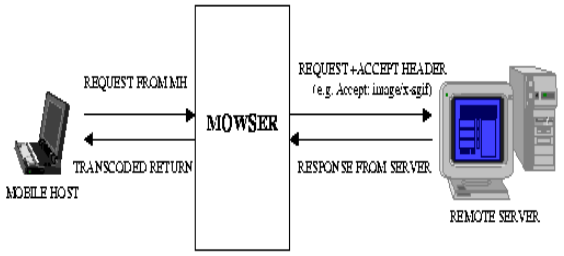
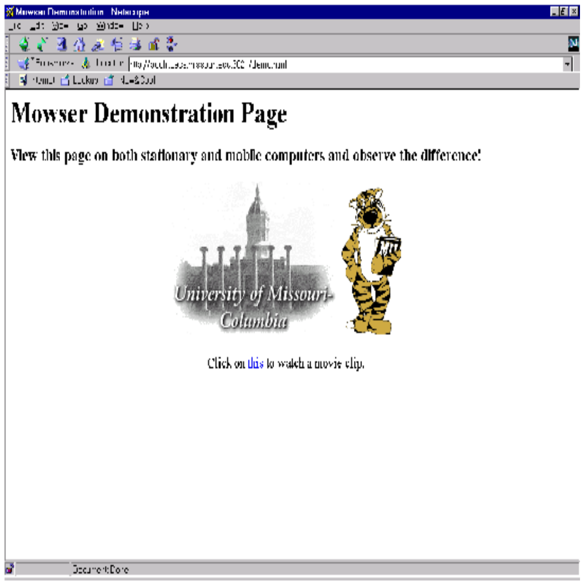


Figure 1. Active Transcoding of HTTP Stream



Mowser Demonstration Page

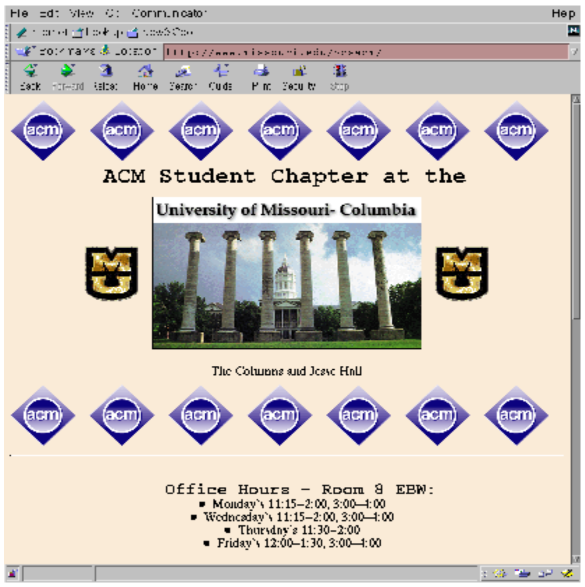
View this page on both stationary and mobile computers and observe the difference!

University of Missouri-Columbia

Click on this to watch a movie clip.

Document: Done

Figure 2. Example of Content Negotiation: Response on a Resource rich Client



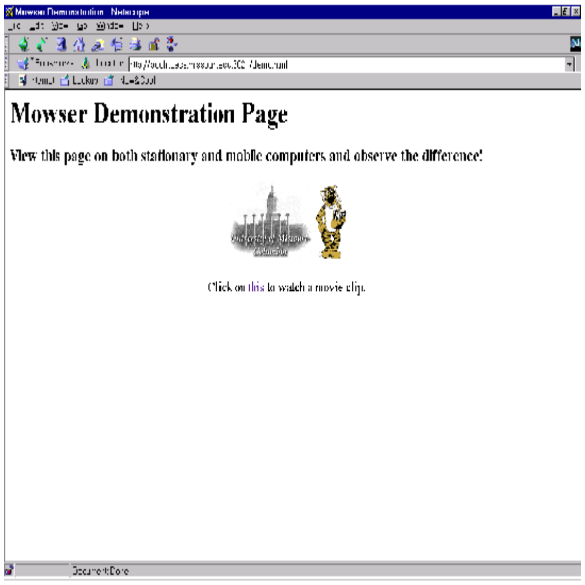
ACM Student Chapter at the

University of Missouri- Columbia

The Columns and Jesse Hall

Office Hours - Room 8 EBW:
- Monday's 11:15–2:00, 3:00–4:00
- Wednesday's 11:15–2:00, 3:00–4:00
- Thursday's 11:30–2:00
- Friday's 12:00–1:30, 3:00–4:00

Figure 4. Example of Image Transcoding: Response on a Resource rich Client



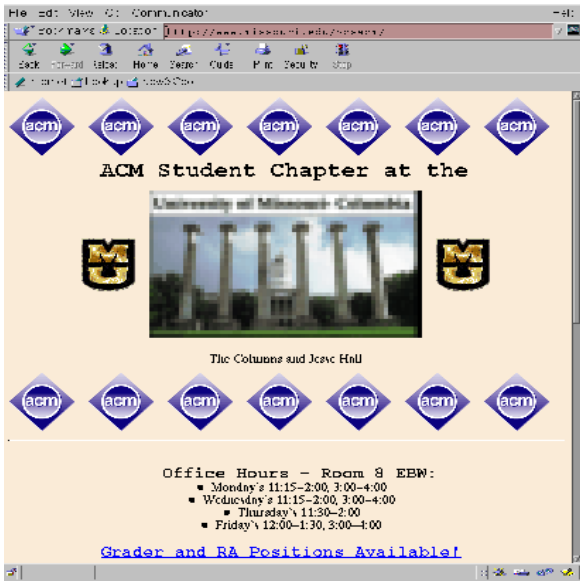
Mowser Demonstration Page

View this page on both stationary and mobile computers and observe the difference!

Click on this to watch a movie clip.

Document: Done

Figure 3. Example of Content Negotiation: Response on a Resource poor Client



ACM Student Chapter at the

University of Missouri- Columbia

The Columns and Jesse Hall

Office Hours - Room 8 EBW:
- Monday's 11:15–2:00, 3:00–4:00
- Wednesday's 11:15–2:00, 3:00–4:00
- Thursday's 11:30–2:00
- Friday's 12:00–1:30, 3:00–4:00

Grader and RA Positions Available!

Figure 5. Example of Image Transcoding: Response on a Resource poor Client