WEB-INTEGRATING NETWORK-CONSCIOUS IMAGE TRANSMISSION

Armando L. Caro Jr.
Paul D. Amer

Computer and Information Science Department
University of Delaware, Newark, DE 19716 USA
Email: {acaro,amer}@cis.udel.edu

Sami Iren

GTE Laboratories Incorporated
Waltham, MA 02451-1128 USA
Email: sami.iren@gte.com

Phillip T. Conrad

Computer and Information Science Department
Temple University, Philadelphia, PA 19122 USA
Email: conrad@joda.cis.temple.edu

ABSTRACT

This paper describes the development of a new Network-Conscious Image Compression and Transmission System (NETCICATS) architecture that may be incorporated into existing web browsers, e.g., Netscape[a]. NETCICATS introduces an approach to compression that does not simply optimize compression, but which optimizes overall performance of transmitting compressed images over a lossy, packet-switched battlefield network. The new architecture includes the development of a Java Application Programming Interface for the Universal Transport Library, a Network-Conscious (NC) image server, a generalized Java applet for displaying all NC images, and an array of Netscape plugins for decoding each type of NC image. Integrating NETCICATS into the web browser framework enhances research collaboration and presents more possibilities for future experiments. [b]

1 INTRODUCTION

Traditionally, research in image compression focused solely on one goal: minimizing the image file size. Current technological advancements in storage media has made saving disk space almost a non-issue in comparison to transmitting images and other multimedia over stressed networks, such as lossy, packet-switched battlefield networks. In our research, we emphasize that optimal image compression algorithms should also account for the network conditions and the Quality of Service (QoS) provided during transmission. To achieve the best performance, algorithms should be optimized to exploit the different QoS that the underlying transport protocols offer.

Network-Conscious (NC) image compression introduces a new concept which takes image transmission into consideration when designing image compression algorithms. Using an Application Level Framing [6] philosophy, an image is compressed into MTU-size[c] Application Data Units (ADUs) at the

[a]Netscape and Netscape Navigator are registered trademarks of Netscape Communications Corporation in the United States and other countries.

[b]Prepared through collaborative participation in the Advanced Telecommunications & Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation thereon. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government.

[c]MTU (Maximum Transmission Unit) is the maximum packet size that a link layer can carry. An MTU-size packet can be transmitted end-to-end over a network without the need for fragmentation and reassembly.

application layer. Each ADU "carries its semantics"; that is, each ADU is a self-contained unit possessing all information necessary for decoding and displaying the information within that packet. Each piece of independent information can be delivered to the receiving application out-of-order, thereby enabling faster progressive display of images [5, 10, 12, 13].

Classical transport protocols that operate over unreliable battlefield networks provide an "all-or-nothing" choice for transport QoS; either total order and reliability (e.g., TCP) or no guarantees at all (e.g., UDP). In a battlefield scenario, using UDP alone without some form of supplemental reliability is unacceptable since its use could result in the loss of some important information. By process of elimination, TCP becomes the protocol used by many applications. However, a total order and reliability protocol restricts the performance of an NC image. NC images can outperform traditionally compressed images only when they can take advantage of out-of-order data.

NETCICATS uses a Univeral Transport Library (UTL) developed within the University of Delaware's Protocol Engineering Laboratory to provide the QoS needs of NC images. UTL is a library of innovative transport protocols that gives application programmers the ability to write to a single API, then experiment with a variety of different transport protocols available. The protocols provide services that can range anywhere between the two extremes of UDP and TCP [4, 8, 9].

Our research demonstrates that with a combination of innovative transport protocols and only a small penalty in compression ratio, today's standard compression algorithms can be modified to provide significantly better overall display of images [5, 12, 13]. Hence, performance in a lossy packet-switched battlefield network can be enhanced. The new NETCICATS architecture will allow others to witness these results on the web.

Section 2 further explains the motivation behind integrating existing web functionality and NETCICATS. Section 3 presents the layout explaining the details of the web-integrated NETCICATS architecture.

2 MOTIVATION

The goal of the new NETCICATS architecture is to have a more portable system which will allow us to more easily demonstrate our ideas and share our work with other researchers. Collaboration will be more effective by giving others not only the opportunity to read our publications, but to also download the software and demo our work. Research becomes more progressive and successful when ideas are exchanged and discussed with others; hence, giving hands-on experimentation to collaborators will clearly enhance the progress of research even more.

Performing a greater variety of experiments is another motivation for having portable systems. In turn, a deeper understanding of our research may be obtained. The new system will provide the additional ability to *easily* experiment over existing network infrastructures which are not self-contained or simulated, such as the Internet.

Client portability, however, is more important than server portability. The idea is that having a portable server is more of a luxury than a necessity. This concept is not unfamiliar to the web's client/server model; once a server is running on a given platform, clients should be able to communicate with the server from multiple platforms. Therefore our work focuses on making the client as portable as possible, with less concern given to server portability.

3 ARCHITECTURE

Figure 1 illustrates the entire web-integrated NETCICATS architecture. The new NETCICATS architecture attempts to create a seamless integration of NC images, HTML, and other web elements. The figure shows specifically the details of how NC-GIF [5, 12, 13] (or NCG) images are handled simply because it is the first image format that Netscape will be extended to handle. However, the system will be nearly identical for all NC images; all the same components will be used, except there will be a different Netscape plugin for each image format. The plugin is the heart of the NETCICATS functionality in that it is in charge of decoding and performing any other image specific tasks.

The entire system operates as follows. When

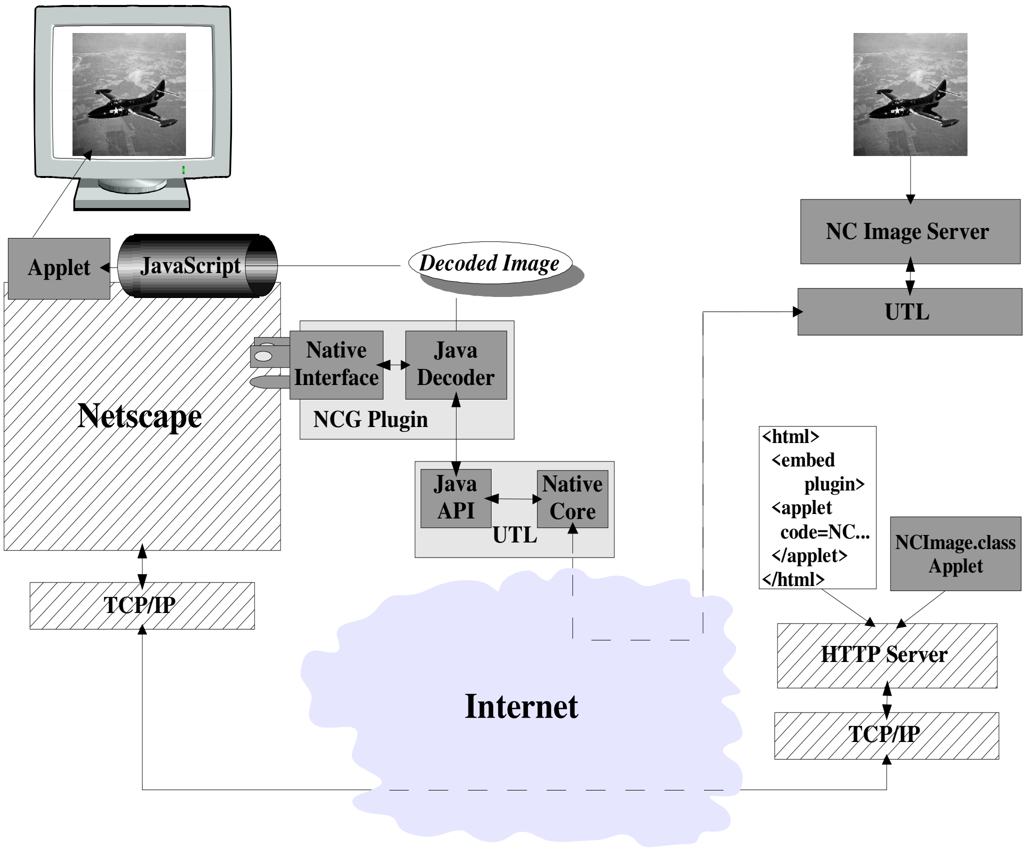

Figure 1: Web-Integrated NETCICATS Architecture

Netscape starts, it checks for plugin modules that have been installed, enumerates them, determines their MIME types, and registers each according to their MIME types. Now the web browser is ready to handle all registered MIME types.

The interaction between the web browser and server remain unchanged; the user specifies the URL of a web page, and the browser retrieves the data over a TCP connection. When the user opens a page that embeds an NC image, Netscape responds with the following sequence of actions (typical for all Netscape plugins) [2]:

- check if the plugin for this NC image type has been installed (i.e., check for a plugin with a matching MIME type)
- load the plugin code into memory
- initialize the plugin
- create a new instance of the plugin

Notice that Figure 1 shows that NC images are not stored on an ordinary web server as traditional images would be, but are instead stored on a UTL-aware NC image server. Because of the specialized location of NC images, a web page must be authored to explicitly specify the entire URL of the NC image location (see the *ncgurl* option in Figure 2). This URL is passed to the plugin during its instantiation phase.

Additionally, the web page must include the general purpose NCImage applet (see Figure 2). The NCImage class is a thin applet that has only one purpose: to provide and maintain the drawing window in which the image will be displayed. Netscape provides a technology called LiveConnect which allows Java applets to communicate with Java-enabled plugins via JavaScript [1]. Our system takes advantage of this technology by using JavaScript to pass down an instance of the NCImage applet to the plugin so that it may draw the image into the applet's drawing

```
<EMBED type="image/x-ncg-plugin"
       name="NcgPlugin"
       ncgurl="pmtp://sauterne.cis.udel.edu:2000:X2/img/air.ncg"
       HIDDEN>

<applet code=NCImage.class name="NCImage"
        MAYSCRIPT height=260 width=260>
</applet>

<form>
   <input type=button value="Get NCG Image"
          onclick='document.NcgPlugin.begin(document.NCImage.thisApplet)'>
</form>
```

Figure 2: Example HTML Code

window (see Figure 2). Once the plugin receives the instance of the applet, it knows that everything has been setup to begin drawing the image. The plugin may now make a UTL connection to the NC image server and begin retrieving the image data. As the encoded data arrives, the plugin may immediately begin decoding and drawing the image on the screen (unlike the case for unordered arrival over TCP).

If a user leaves the instance's page or closes its window, the plugin instance associated with that page is deleted. When the last instance of a plugin is deleted, the plugin code is unloaded from memory. Other than disk space, plugins consume no resources when not loaded [2].

3.1 AN NC IMAGE PLUGIN

An NC image plugin consists of two parts (refer to Figure 1): the *Native Interface* and the *Java Decoder*. The Native Interface is a dynamic load library written in C which is a thin module that is solely responsible for interacting with Netscape during plugin registration, loading, initialization, instantiation, and shutdown.

Netscape's LiveConnect technology allows a plugin to have a Java counterpart which can interact with the native C code [1, 11]. In our plugin, the Java counterpart makes up the core functionality. The Java Decoder is responsible for making a UTL connection to the NC image server, retrieving the encoded image data, decoding it, and passing the raw pixel data to the applet to be displayed. Since the applet, and not the plugin, is responsible for displaying the image, the *HIDDEN* option should always be used as shown in Figure 2. Implementing most of the plugin in Java gives us more portability for our NETCICATS clients; as explained in Section 2, this is our goal.

3.2 UTL JAVA API

UTL is a C library that provides a variety of transport protocols through a single API [4, 8, 9, 7]. However, since the NC image plugins were primarily written in Java, the ability to use UTL in Java applications became a necessity. There were two possibilities: (1) completely rewrite UTL in Java, or (2) create a Java API that would call UTL functions using the Java Runtime Interface (JRI)[d] [11]. Rewriting UTL in Java is a big task and would further delay the completion of the web-integrated NETCICATS. To obtain a working system more quickly, the Java API solution was used. In the future, a Java implementation of UTL may be developed to make NETCICATS clients more portable.

4 CONCLUSIONS AND FUTURE WORK

The new web-integrated NETCICATS architecture has been described. Most of the components can be generically used for all NC images; only the plu-

[d] JRI is a technology developed by Netscape which is very similar to Sun Microsystem's Java Native Interface (JNI) technology. It allows interaction between C and Java code. However, JRI may only be used with Netscape's Java Virtual Machine.

gin is image specific. Currently, the plugin for NC-GIF images is being completed. In the future, the NC-SPIHT [5, 12, 13] plugin will also be implemented. However, to compare their performance against their classical compression counterparts in a fair manner, the counterparts must also have a plugin that uses the same transmission system. For example, we must implement a standard Gif[e] [3] plugin that retrieves the image data from a NC image server over a UTL connection. In this way, any additional delays and artifact introduced by NETCICATS will be equally accounted for on both sides of the comparison.

Currently, the NC image server is a simple piece of software. It listens on a port for requests and responds by transmitting the requested image data. It does not support multiple, concurrent connections. In the future when the system is complete, there will be a need for such support. We will want to allow users to have the ability to retrieve multiple images from our servers simultaneously. With this ability, web pages can be authored to do side-by-side comparisons of various image compression and transport protocol combinations.

Section 3.2 explains that a UTL Java API has been developed to allow NETCICATS to be web-integrated quickly. To increase portability of the client, a Java implementation of UTL should be written. In fact, without one, the client is not very portable due to its dependence on UTL. Implementing nearly all of the plugin functionality in Java makes the plugin itself portable, but it will be useless on any platform which UTL has not been ported to.

Java is notorious for having slow performance. Hence, it will be additionally interesting to see the performance of transport protocols written in Java.

5 DISCLAIMER

[e] GIF is a Service Mark of Compuserve, Inc., Columbus, OH.

6 REFERENCES

[1] LiveConnecting (Netscape) Plug-ins with Java. `home.netscape.com/eng/mozilla/3.0/handbook/plugins/pjava.htm`.

[2] (Netscape) Plug-in Guide. `developer.netscape.com/docs/manuals/communicator/plugin/index.htm`.

[3] Graphics interchange format, version 89a. Technical report, Compuserve Incorporated, Columbus, Ohio, July 1989.

[4] P. Amer, C. Chassot, T. Connolly, M. Diaz, and P. Conrad. Partial order transport service for multimedia and other applications. *IEEE/ACM Trans on Networking*, 2(5):440–456, October 1994.

[5] P. Amer, S. Iren, G. Sezen, P. Conrad, M. Taube, and A. Caro. Network-conscious GIF image transmission over the Internet. *Computer Networks*, 31(7):693–708, April 1999.

[6] D. Clark and D. Tennenhouse. Architectural considerations for a new generation of protocols. In *ACM SIGCOMM '90*, pages 200–208, Philadelphia, PA, September 1990.

[7] P. Conrad. Order, reliability, and synchronization in transport layer protocols for multimedia document retrieval. PhD Dissertation, CIS Dept. University of Delaware, (in progress).

[8] P. Conrad, P. Amer, E. Golden, S. Iren, R. Marasli, and A. Caro. Transport qos over unreliable networks: No guarantees, no free lunch! In *IFIP Fifth International Workshop on Quality of Service (IWQOS '97)*, New York, NY, May 1997.

[9] P. Conrad, P. Amer, M. Taube, G. Sezen, S. Iren, and A. Caro. Testing environment for innovative transport protocols. In *Advanced Telecommunications/Information Distribution Research Program*, College Park, MD, February 1998.

[10] C. Diot and F. Gagnon. Impact of out-of-sequence processing on data transmission performance. *Computer Networks*, 31(5):475–492, March 1999.

[11] Warren Harris. The Java Runtime Interface. `home.netscape.com/eng/jri/index.html`.

[12] S. Iren. Network-conscious image compression, 1999. PhD Dissertation, CIS Dept., University of Delaware.

[13] S. Iren, P. Amer, A. Caro, P. Conrad, G. Sezen, and M. Taube. Network-conscious compressed image transmission over battlefield networks. In *Advanced Telecommunications/Information Distribution Research Program*, College Park, MD, February 1998.