Sunil George
Columbia Video Network - Advanced Internet Services
Columbia University
NYC, NY
SGeorge@na2.us.ml.com

Introduction:

Currently, the site management software primarily produce a graphical representation of a web site, indicate how many broken links, and where the broken links are located. Information on when the page was last modified and the advanced web features is not provided by the site management software packages. The advanced web features for this project are tables, applets, frames, objects, cascading style sheets, and backgrounds.

I decided not to determine whether a page is cacheable, if there is correlation between age of a page and broken links, or even try to apply Zipf's law because as a webmaster these things are not important to me. As a webmaster of a single web site, I am more concerned with broken links, updating web pages, security, navigation of a web site, and organizing a web site in a way that enables the user to quickly find the desired information. A network administrator or someone in charge of a proxy server would be more interested in whether a page can be cached or not, because of concern for network bandwidth and response time. The correlation between age of a page and broken links will lessen in importance because more web pages are being dynamically created by databases. Zipf's law is more academic than business related.

I approached designing the program from the point of view of analyzing pages from a single web site. Second, I chose Java because it simplifies network programming better than C or C++ and because of its cross platform capabilities.

Most Difficult Problem To Solve: Java gives you the ability to read the input stream either as individual characters or each line in the input stream as a string. The BufferedReader class gives you the ability to read each line in the input stream as a string. The code to give you the ability to read each line as a string is

```
BufferedReader in=new BufferedReader(new InputStreamReader(instream.getInputStream()));.
```

The above line of code will not work in Macintosh version of JDK 1.1. The reason is that the BufferedReader and InputStreamReader classes are not found in the Macintosh version of JDK 1.1. Because I was developing my program on the Mac, I had to find another way read an input stream into a string. Compounding the problem was that the integer representation of each individual character was being returned by the read method that read the input stream character by character. Below is the actual piece of code that I used to solve the problem.

```
InputStream datain=http.getInputStream();
while ((c=datain.read()) !=-1)
{
        x=(char) c;
        ch=new Character(x);
        temp=temp+ch;
}
```

Sunil George
Columbia Video Network - Advanced Internet Services
Columbia University
NYC, NY
SGeorge@na2.us.ml.com

Architecture:

What it does: The program analyzes an entire web site to determine how many of HTML's advanced features are used, when the pages were last modified, how many internal and external links, and how many broken internal links there are.

What it doesn't do: The program does not check for duplicate internal links and does not open non-Http protocols. An example of a non-Http protocol is FTP.

Screen: The user interface consists of two text boxes, a text area box, a button labeled "Click Me To Begin," and a title. The topmost text box is where the user types in the web address. The text box is pre-filled with the phrase Http://. The second text box will show the URL of the Web page that is being analyzed. The text area box will show the result of each Web page analysis.

Functions: After the "Click Me To Begin" button is clicked, the buttonBegin_ActionPerformed function is executed until all the URLs are analyzed. It can call up to two functions, DoNumbers and LinkSize, to analyze the web page indicated by the URL and get the next URL depending upon whether the URL is internal or external. Only internal URLs are analyzed. The DoNumbers function is where the actual analysis of the web page is done. It takes as a parameter the URL of the web page that will be analyzed and print out to text area box the result of the analysis. The functions called by DoNumbers are ReadPage, LoadLinks, TableCount, FrameCount, BGCount, SSheetCount, AppletCount, and ObjectCount. Each of the "Count" functions analyzes the string returned by the ReadPage function, to determine whether a particular HTML advanced feature is present on the web page and increments a counter if the feature is present. The ReadPage function retrieves web page's HTML code and loads it into a string. The LoadLinks function loads a URL onto a list of URLs. The LinkSize function retrieves the next URL from the URL list. After all the URLs are processed, the final count is printed to the text area box.

Documentation:

Java Classes: AboutDialog, ExitDialog, and WebReader. The WebReader class is the main Java class that executes the program. The AboutDialog class produces an About dialog box when the user selects the About menu item found under the File menu. The ExitDialog produces a dialog box that gives the user the option to exit the program or not. The ExitDialog box appears when the user selects the Exit menu option found under the File menu.

Future Enhancements: Change the program to analyze each web page and document each HTML feature that is present on each page. And store the results in a database. For example, let's say a web page had an image tag (<img>) on it. The information stored in the database about the image tag would be name, source, height, width, and what page it appears on. Essentially, I want to change the

Sunil George
Columbia Video Network - Advanced Internet Services
Columbia University
NYC, NY
SGeorge@na2.us.ml.com

program to do full documentation on web pages so that a programmer could easily figure out what each page has, if the page has to be modified later.

Requirements: JDK 1.1.5 preferred.

Development Platform: Macintosh Power PC and Symantec Visual Café 2.0 Professional Development Environment for Macintosh.