Timed Links Solve the "Stale URL" Problem

David Stotts
Department of Computer Science
University of North Carolina at Chapel Hill
`http://www.cs.unc.edu/ stotts/`

Stotts and Furuta presented a paper at Hypertext '91 on how to use the timed links in Trellis to manipulate the apparent structure of a hyperdocument [1]. From the questions and discussion at the conference, I concluded that the community largely did not see a pressing need for either timed links or the manipulations described in the paper, which included selectively hiding content elements and selectively hiding links.

Timing is here to stay, however, in the current release of the Netscape Navigator; and, interestingly enough, timing offers a solution to a serious problem many are finding in the Web. For lack of a better name, call it the "Stale URL" problem: when a Web page is moved (or when file/directory names are changed) any URLs to the original page that remain "out there" in other pages effectively point to missing information. Based on the results in [1], we demonstrate here how to use the timing feature of Netscape to solve the "Stale URL" problem. This solution is controlled in a decentrallized fashion, by individual authors, without the need to keep up-to-date a central relocation file for the server.

In the following discussion we will refer to the URL that has gone stale as *StaleURL*, and we will designate the location to which the information has been moved as *NewURL*. The solution commonly encountered today is manual indirection. A small HTML file is left at *StaleURL* saying "We've moved... please go to *NewURL*." The reader is expected to manually select the link for *NewURL* to obtain the original information.

Colleagues have suggested potential solutions involving placing a simple script at the location *StaleURL*. The idea is to have the script run when *StaleURL* is requested and cause a server-side include of *NewURL*. The attractiveness of this approach is that the reader never knows *StaleURL* is stale – the ostensible behavior is exactly what one expects from retrieving a "fresh" URL. Unfortunately, for security reasons Web servers cannot be convinced to run scripts found in arbitrary locations.

The solution using Netscape Navigator timing actually simulates the manual indirection method, but without user intervention. The $< meta >$ tag can be placed in the header of an HTML file, and Netscape can be instructed to refresh the page every so often. Consider a file named "ThisFile.html" containing this header:

```
<meta http-equiv="Refresh" content="9; URL=ThisFile.html">
```

Netscape Navigator will reload the file every 9 seconds, with no user intervention required. Given this behavior, the key to automatic URL redirection key, is to set the refresh interval to 0 seconds and set the URL to point to *NewURL*. At *StaleURL* we place an HTML file as simple as this:

```
<head>
<meta http-equiv="Refresh" content="0; URL=NewURL">
</head>
<body> ...  </body>
```

Now, when a browser requests *StaleURL*, this file is returned. If the browser is Netscape, it recognizes the refresh timing in the $< meta >$ tag and immediately initiates a retrieval of *NewURL*. While we have presumed this to be the new location of the information in question, automatic redirection can be cascaded to get many levels of indirection.

It obviously doesn't matter what the body content is, since the reader is not given any time to view this file before the contents of *NewURL* are rendered. However, in case *StaleURL* is retrieved by a browser without timing, it seems a good idea to put the traditional "We've moved..." message in the body anyway.

Drawbacks

A few points deserve some discussion. First, the timing feature currently in Netscape is actually a timing value on a content page, not on individual links, so it does not provide the complete solution described in [1]. In effect, it serves as a common timing definition for any link pointing to that page; all links to any one page will have the same timing value. A more general solution would allow two different links pointing to the same page to have two different time values; this in turn would allow a page to be redirected in one document, but not in another.

Secondly, this solution presents a small problem when backing up. If a reader has been redirected from a source page (containing the *StaleURL*) to a destination page, but has gone through a redirection page, the desired semantics for "go back" would be to return to the source page. With the timing solution described here, backing up takes one to the redirection page; due to the timing feature, the reader is then immediately sent again to the destination page again. One way to really return to the source page is to pull down the history list and go back two (or more) pages directly. Another is to make the timing on the redirection page something longer than 0 seconds, say, 1 or 2 seconds. Then the reader has a brief moment when backing up to back up a second time to the original source page.

Access this article, and try for yourself the concepts explained here, at

`http://www.cs.unc.edu/~stotts/staleURLdemo/`

References

[1] P. David Stotts and Richard Furuta. Dynamic adaptation of hypertext structure. In *Proceedings of Hypertext 91*, pages 219–231. ACM, December 1991.