Brock University

Department of Computer Science

Subject-Based File-Link System for the Web

Tomoharu Arakawa and Jerzy A. Barchanski
Technical Report # CS-03-05
April 2003

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

Subject-Based File-Link System for the Web

Tomoharu Arakawa, Jerzy A. Barchanski
Brock University
St. Catharines, Ontario
Canada

Abstract

The system described in this paper is a kind of directory system designed for organizing files on the Internet based on subjects and for sharing information among a group of people. The system consists of a server program and a client-side browser. The server program is a module of an Apache web server while the client-side browser is a web page with functions enabled by JavaScript. The system creates XML files, each of which acts as a node in a logical tree structure and stores them on the server. The node represents a directory or a file of the tree structure. The system can be used by a small to medium group of people who want to share an information specific for the group or it can be used to organize web documents as a private directory collection.

1. Introduction,

Many users of the Internet perceive it as a large collection of information - a kind of on-line encyclopedia. It stores an enormous amount of almost every category of information. To make it available, search engines and directory systems were developed. While a lot of research was done on faster and more efficient search algorithms, the methods of organizing and sharing information have not been greatly improved.

We describe in this paper a system which lets a group of Internet users create a shared directory, submit their own documents to it, manipulate it and search it. Although there are already similar systems on the Internet, they separate responsibilities of the two parties - system maintainers and end users. The difference between our system and the other systems is that any user of our system can maintain his/her own archive while in the others only the owners or administrators of the servers can do it.

Our system, called mod_filelink (filelink in the rest of the paper), can be used by any Internet user to organize files on the Internet and to share information among a group of people with a common interest. It can be used as a personal bookmark file, or as a tool to archive information shared by a group of people who know each other, such as employees of a corporation.

Although the filelink system is designed for anyone who knows how to create a simple web site, it can be used only by users of a UNIX/Linux machine that uses Apache [1] for its web server. Since filelink is an Apache module [3] , the installation of the module should be done by the web server administrator. If you are a system administrator of a UNIX/Linux machine that runs the Apache server, there is no problem. If you are a user of a UNIX/Linux machine, you have to ask your system administrator to install the module.

The above restrictions on filelink installation apply only to a filelink archive owner - i.e. a person who creates and maintains a filelink tree. Filelink trees created by the filelink archive owners can

be accessed by any user with the Internet Explorer browser. You can consider therefore the filelink archive as a kind of web site.

2. Survey of similar Internet directory systems.

2.1 Open Directory Project

The largest directory project in the world is the Open Directory Project [6]. It is the largest, most comprehensive human-edited directory of the web. It is constructed and maintained by a large global community of volunteer editors. Many people probably don't know about the project, but in fact, the Open Directory powers the core directory services of the web's largest and most popular search engines and portals, including Netscape Search, AOL Search, Google, Lycos, HotBot, DirectHit, and hundreds of others.

The Open Directory data is made available for free to anyone who agrees to comply with their free-use license. There is no cost to submit a site to the directory, and/or to use the directory's data. The Open Directory is hosted and administrated by Netscape Communication Corporation. They operate the directory by a small staff responsible for editorial policies and direction, community management and development and systems engineering. Other directories such as Yahoo! and LookSmart are developed and managed by a small paid staffs. The Open Directory Project was founded on the assumption that directories with small staff can not scale well to the growth of the web, and still maintain a quality, current directory. It is developed and managed therefore by a growing community of Internet users who are experts in some areas of interest. The community accept applications for new volunteer editors who select, evaluate, describe and organize web sites. They are responsible for reviewing submissions to their categories, and list sites according to prescribed editorial guidelines.

The Open Directory has a submission page, through which anyone can submit a URL with the title and description of his/her site. It is easy, but it requires following provided guidelines. It will be reviewed first by one of the Open Directory editors. You can add a site but you can not create a new directory. Added URLs are site-based, not page-based. Even though you have to be aware of these rules, adding a site is very simple and easy - you navigate yourself traversing directories, and as you find the most appropriate category, click a link 'add URL'.

2.2 WebSight Directory System

The most similar to our system is the WebSight Directory System [5]. In terms of structure and method of page submission it is similar to the Open Directory. Submitted pages are checked first by editor(s). That means that they may not be immediately available in the system. The search function returns a well formatted result that allows a user to access the pages found as well as their categories. The search covers only titles and descriptions stored, not the content of linked pages. Websight is well designed and portable, but its maintenance is restricted to system administrators. It uses mySQL for its data storage. While mySQL has a reputation that it provides fast operations, we don't like the idea of accessing a single central database whenever the server receives a request from a client. Once the database gets large and the number of accesses increases, it will create extra overhead and traffic conflicts. Even though writing simple SQL queries is not difficult, the

system does not allow users to modify queries. Because of this, using one of the SQL servers is not appropriate.

3. Representation of directory hierarchy

Hierarchical structures are very common in computer systems and are used in many places. Computer users are familiar with these structures since they are used to create directories to categorize different kinds of files and to store them separately. To get a particular file, the user needs to give its path name or to traverse the directories hierarchy. If you had a small number of files you probably would not care how to get one, but with a large number of files we need an efficient way to do this. Introduction of graphical user interface improved this dramatically, so now we don't have to type path names to reach a file.

For large and complex systems it is always desirable to be able to see them as a whole and as a composition of its parts. We wanted therefore to create such a hierarchical directory structure that allow users to move easily between higher and lower levels of details. Example of such a hierarchical directory structure is the Mac OS X's Finder interface illustrated in Fig.1 below.

This is one of three views it provides.

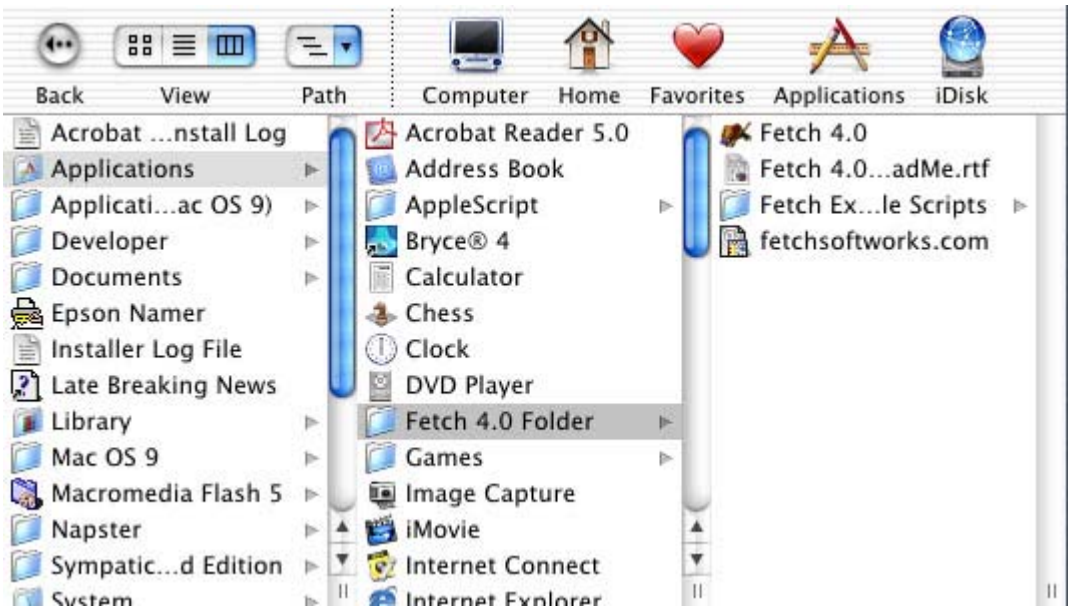


Fig.1. Graphical interface of the Mac OS' Finder directory system.

Each column can be seen as a directory or a folder, and files and folders in the column are what the directory holds. A single click on a folder cell displays its content in the right, next column. In this example, you can see three depths at the same time. The direction to the left represents abstract (shallow in the tree) and the direction to the right represents detail (deep in the tree). When you use this interface, you normally have 3 - 5 visible columns, so that you can see 3 - 5 depths at the same time. This representation of hierarchy clearly shows the directions of abstract

and detail, and allows the user to move in horizontal directions. Vertical movements are limited within the columns, so it does not distract the level of abstraction.

Other popular hierarchy representation used e.g. in MS Windows directory systems is shown on Fig.2.

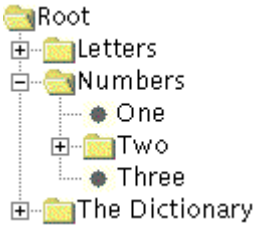


Fig. 2. Graphical interface of the kind used by the MS Windows directory system.

When you click on a folder, the already-visible items below the folder have to slide down in order to squeeze new items. If you continue this operation, it soon goes out of bound, and you have to scroll the whole tree to see the items you already saw. We consider the Mac OS' representation more convenient for users so we adopted it in our system.

4. Overview of the mod_filelink system

4.1. General structure

The filelink is a module developed for Apache 1.3.x, that creates and manipulates logical hierarchical structures on the server. The primary purpose of the module is to logically organize files on the Internet based on subjects. The module uses an XML [8] file to represent a node. The node may acts as a directory holding other documents' information or a file referring to an existing page on the web.
The nodes (XML files), in consequence, compose a logical hierarchy as shown on Fig.3 below.

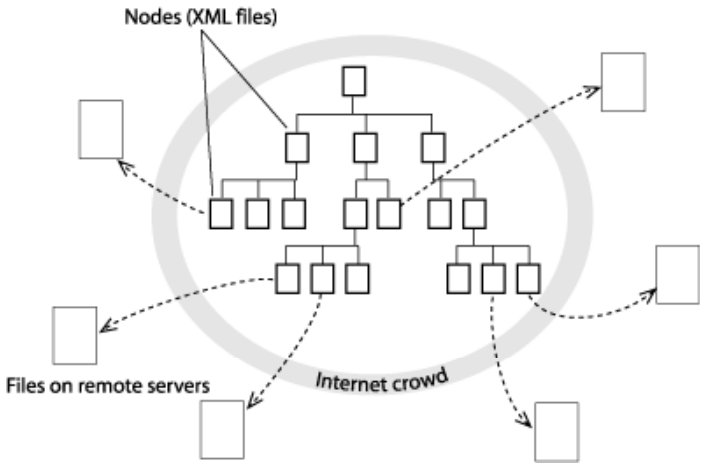
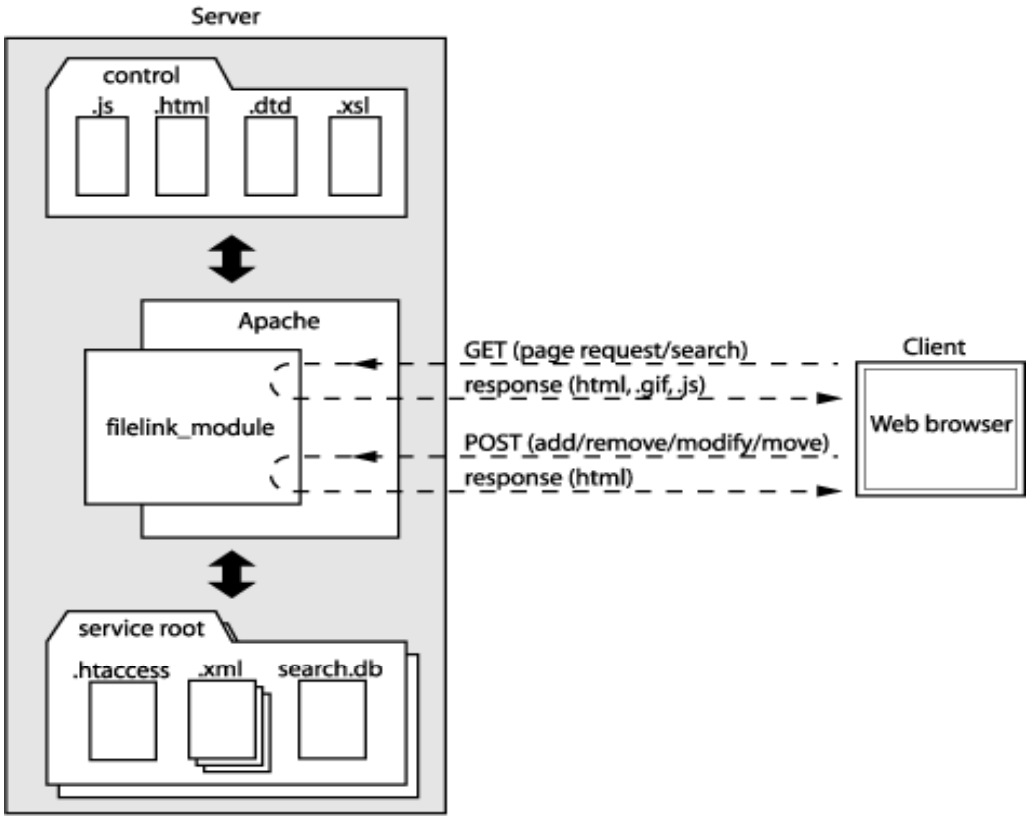


Fig.3. General structure of the filelink system

The filelink archive is shown here as a tree structure even though all XML files for the structure are stored in a single directory on the server. The reference to an outside file is a simple hyperlink after the XML file is transformed into HTML. The current version of the module handles this transformation even though it might not be necessary doing so on the server side, and it may be changed in the future.
The nodes (XML files), in consequence, compose a logical hierarchy.
Once the module is set up properly, users of an UNIX/Linux system can create filelink structures by providing designated directories (one for each structure) anywhere on public-accessible paths. The information for a single node is stored in a single XML file. If you have, for example, 100 nodes in an archive, you have 100 XML files stored in the same directory on your machine. Each of these files has a unique and arbitrary file name. Manipulations (add, remove, modify and move) of the node can be done through a client-side browser by those having permission. Since each node is just a plain XML file, it has to be transformed into another format in order for users to view the file's content in a reasonable style. Writing an XML file is done based on data submitted by the client and a DTD file provided at set up. Transforming XML files (basically into HTML code) is done based on an XSL file also provided at set up. From clients' point of view, the system is akin to Yahoo! or the Open Directory Project. When the client accesses a filelink archive site using a web browser, the module responses with a DHTML page that allows the client to traverse the directories to find particular categories, or to search files by keyword. But unlike those large directory systems, which store all information as data on one system, the filelink system can be anywhere and owned by anybody. It is possible to have multiple archives by one owner, but all archives created by filelink are independent of each other. The filelink archive can be publicly accessible or restricted to those the archive owner allows.

4.2. Computational model and its components

Fig.4. below shows the client-server model used in the filelink system.



The server side is a UNIX/Linux machine running Apache 1.3.x with the filelink module. The module accept GET and POST http methods [4] to one of service roots recognized by the module. The service root is a directory that the owner of the archive sets by the FilelinkServiceRoot directive in the .htaccess file in the same directory. The path to the directory, in consequence, turns out to be the URL of the archive. By default, the service root stores XML files created by the Add requests from the client. Each of these XML files represent a node in the logical hierarchy, and no other information is necessary to form the tree structure. Therefore, the filelink archive can be seen as a collection of these XML files.

4.3. Control Directory

The control directory is specified by the FilelinkControlPath directive, and normally contains DTD, XSL and other files used for the client-side browser. By default, this directory is set as DocumentRoot/mod_filelink, where DocumentRoot, specified in Apache's httpd.conf, is the directory out of which the server serves documents. Thus, by default, there is only one control directory in the server. It can be overridden however by the directive, and users are free to create other control directories with necessary files. Note that the control directory must be executable and files in it must be readable by public.

There are two DTD files in the control directory - one for nodes in the logical tree and the other for the list presentation used for search results. The directives FilelinkXMLFile and FilelinkXMLListFile are used to specify alternative DTD files.

There are also two XSL files in the control directory - one for transforming nodes in the logical tree and the other for transforming the XML document as a result of the search operation. The directives FilelinkXSLFile and FilelinkXSLListFile are used to specify alternative XSL files.

4.4. Service Root

The service root is a directory that turns out to be the archive's URL. It can be specified by the FilelinkServiceRoot directive in any .conf files (httpd.conf, srm.conf, access.conf) or htaccess files. Since ordinary users in the UNIX/Linux system don't have a permission to access those .conf files, their choice is only one - htaccess file. By default, it is also the directory in which all XML files and the search database file (search.db) are stored. The service root directory must be writable and executable by public.

4.5. Client-side Browser

When a client accesses one of the filelink archive using a web browser, the module returns a DHTML page that works as a client-side browser. The browser provides several functions with which the client is able to access or manipulate the archive. For example, the client can traverse the hierarchy, which the archive provides or search pages by keyword. The client is also able to add, remove, modify or move nodes if he/she is allowed to do so.

Navigation of the web is very simple and intuitive, but the mechanism behind it is very complicated. The page is composed of 12 HTML files, 10 GIF files and 1 JavaScript file. The JavaScript consists of 32 functions in about 700 lines. The browser does as much as possible to remove the unnecessary requests over the network. The browser consists of three panes. It provides function buttons on the top pane, a tree or search view in the middle pane, and an extra

submission-purpose page in the bottom pane.

The browser provides a tree view which is similar to Max OS X's Finder. There are four frames in the middle pane by default, and the number can be changed between 3 and 6 by changing the Frames pull-down menu. Each frame contains one HTML-transformed XML file, representing a single node. Clicking on an item in a frame opens the specified page on the right, next frame. Unless a new node goes out of bound, loading an XML file does not affect other parts of the browser. A variable, that is stored in the index page (mod_fl_index.html), holds the frame information and a list of node pages that forms a branch of the logical tree from the root node to the currently selected node.

The default browser is written in HTML and JavaScript, and it works on Internet Explorer. It does not work yet properly on other web browsers.

Fig.5. Top pane of the Filelink browser

The browser has a navigation bar in the top pane. It looks like the one shown on Fig.5.: There are two modes in the browser - tree and search. Switching between them is as simple as clicking one of two tabs, named Tree and Search on the browser. The two modes vary in how they represent pages in an archive.

a. Tree mode.

In the Tree mode (Fig.6), the tree tab is red and the middle pane is split into three or more frames (four in default). Each frame is a page that works as a directory or a file reference, representing a node. A file reference contains a hyperlink to a web page somewhere on the Internet. A directory may contain subdirectories, showing them in separate cells, row by row. If you click one of the cells (subdirectories), it will be highlighted and will display its content in the right, next frame

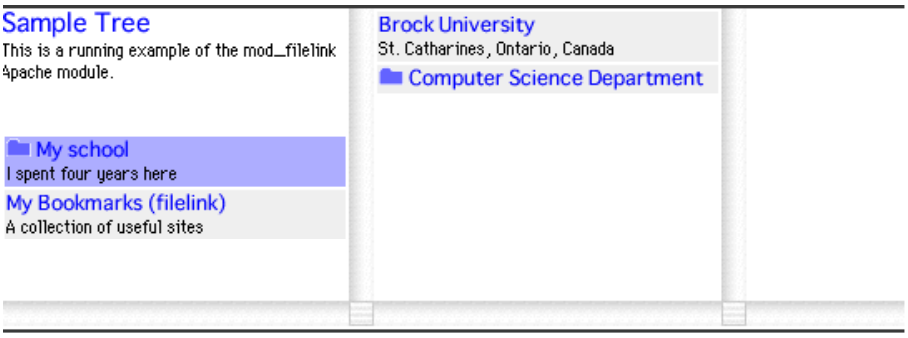
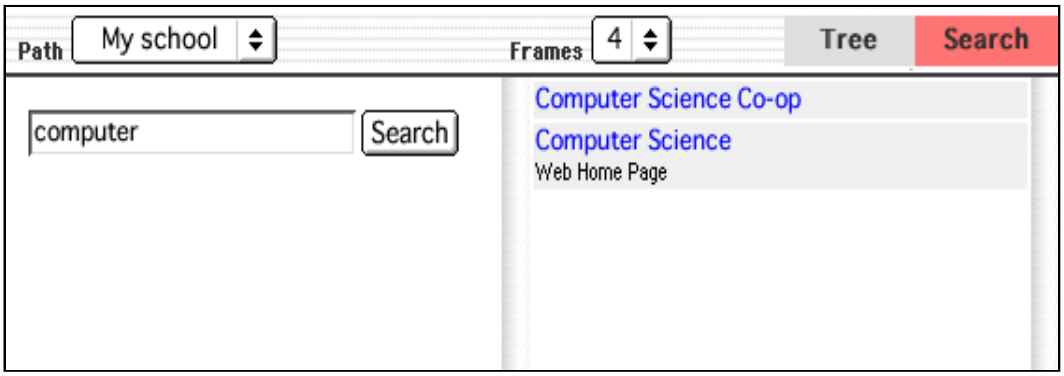


Fig.6.

Window of the Tree mode;

b. Search Mode



In the search mode (Fig.7), the 'Search' tab is red, and you can conveniently search documents by keywords.

Fig.7. Window of the Search mode.

Even though filelink is a well-organized archive, someone may want to get directly a subject he/she is looking for. When you click the 'Search' tab, the middle pane is replaced with another frame that contains a field for keyword entry. Submitting a keyword will request the filelink module to search any pages that contain the keyword, and the module returns a result into the right frame if any. The result is formatted so that it is similar to a directory in the tree mode, showing pages row by row.

The module uses the UNIX database to store keywords and corresponding file names for the search operation. Any words entered through ADD or MODIFY requests are stored as keywords. The current version does not look into web pages hyper-linked from nodes in the filelink archive. Clicking (selecting) on one of the rows that contains a page title found by the

search highlights the cell. Then, you can click the 'Tree' tab to switch back to the tree mode. In this case, previously opened pages in the frames are gone and the browser loads new pages to make a path to the page from the search result. If a page found by the search is a directory, double-clicking the cell does exactly the same thing as selecting the cell followed by clicking Tree tab. If the page is a file reference, double-clicking the row will pop up a new window and show the referenced page in the window.
You can enter multiple keywords putting a white space between them.
The module takes all keywords separately, searches the archive by all keywords, and then it performs logical AND operations on all matches. The result contains therefore only pages that satisfy all keywords.

Since the archive is organized in a hierarchy, page found by a search request is considered a path rather than an individual file. That is, the page has the information of all ancestor directories it inherits.

5. Conclusions

We have described in this paper a kind of directory system designed for organizing files on the Internet based on subjects and for sharing information among a group of people. The system consists of a server program and a client-side browser. The server program is a module of an Apache web server while the client-side browser is a web page with functions enabled by JavaScript. The system creates XML files, each of which acts as a node in a logical tree structure and stores them on the server. The node represents a directory or a file of the tree structure.

The system works fine as a small directory system and provides both archive maintainers and end users an easy access. We plan to further improve the system by internationalization, adding a custom log feature and speeding up the client-side browser.

References:

1. "Apache HTTP Server Project." The Apache Software Foundation. 20 Jun. 2002 <http://httpd.apache.org/>
2. Bates, C. (2000). Web Programming: Building Internet Applications. New York: John Wiley & Sons, Inc.
3. Dunkel, Philipp. "mod-xslt Apache Module." 20 Jun 2002 <http://www.mod-xslt.com/>
4. Krishnamurthy, B. , & Rexford, J. (2001). Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement. Massachusetts: Pearson Education, Addison-Wesley.
5. Koopmanschap, Stefan. "WebSight Directory System." SOURCEFORGE.NET 20 Jun. 2002 <http://websight.sourceforge.net/>
6. "Open Directory Project." Netscape. 20 Jun 2002 <http://dmoz.org/>
7. Stein, L. , & MacEachern, D. (1999). Writing Apache Modules with Perl and C. California: O'Reilly & Associates, Inc.
8. Veillard, Daniel. "The XML C library for Gnome - libxml." 20 Jun. 2002 <http://xmlsoft.org/index.html>