An Environment for Mirroring Hypermedia Documents

László Kovács <laszlo.kovacs@sztaki.hu>
András Micsik <micsik@sztaki.hu>
Gábor Schermann <schermann@sztaki.hu>

Abstract

Mirroring (creation of remote copy) of WWW hypermedia documents is discussed. First a Portable Hypermedia Document (PHM) format is defined. Mirroring service and a software environment for mirroring are introduced. Mirroring environment automatizes the process of mirroring and transforms WWW hypermedia documents into PHM format. The algorithm and the architecture of the environment of mirroring service are described in detail.

I. Introduction

The World Wide Web (WWW) is a networked hypermedia architecture [1], [4], joining millions of documents together nowadays via hypertext links. Documents are stored on server machines and client software running on practically any kind of networked computer is used to retrieve documents through Internet.

Mirroring in the Internet jargon means the creation of a remote copy for some data or complete hypermedia documents. This technique is used for information that is very popular or served via low-speed connections. It can help to decrease network traffic over the Internet backbone. Various techniques of mirroring work well for other types of Internet services such as FTP [9] or USENET News and have enormous significance in the area of World Wide Web that generates most traffic of all services over the Internet. Although there exist a few public domain scripts for WWW mirroring [10], the topic is in a somewhat premature state according to the evolving needs of the society of Internet. This can be caused by the spread of WWW caches.

I.A. Caching versus Mirroring

Caching on Internet is similar to disk caching on computers. Caches serve the recently viewed documents to avoid repeated download of the same data. The following table shows the differences in functionality between caching and mirroring.

Caching is an on-request dynamical technique whilst mirroring is static, request-independent manner of document access speedup. In case of caching document is downloaded at the time of the first request. The second and next requests are served locally. In mirroring, the documents are downloaded before any requests and requests (including the first) are locally served.

Caching generates a low profile, permanent traffic. Mirroring generates "burst-like" traffic, short heavily loaded periods and long, silent periods. Internet night traffic is usually much lower than in daily hours, mirroring could make less traffic and less trouble in daytime.

Mirroring downloads all coherent documents, caching doesn't. In this case mirroring is useful when the documents are big and/or the connection is too slow to download documents on request.

If documents are downloaded in the mirroring scheme, data are persistent. Caching has temporal documents, if the cache-buffer is filled, it will throw out some data. Mirroring makes you sure to have the document locally.

Under mirroring the allocated disk space is static, not changing. Under caching this size is dynamically changing. There are various cache-techniques, utilization of disk space is heavily depends on it.

The goal of using caching is an overall performance improvement. Mirroring is frequently used when fast access is needed on some given documents.

These two techniques can be used together, because they speed up different types of requests. For browsing the Internet, caching is a better choice. But if something important is needed to access at any time, mirroring is necessary, because only mirroring ensures the documents to have locally. For example, if some programs or programming languages have descriptions, FAQs, tutorials, etc. in HTML-format, and you want to use that, it is simpler to mirror documents in a night hour than downloading document one-by-one on requests.

As a result one can conclude that caches do not copy WWW documents completely, and therefore caches are not able to provide a true secondary location of a document with a high degree of availability. For this one need solutions for mirroring. The process of mirroring should be an intelligent and automatic task. As a side effect the exchange of complex HTML documents over the Internet would be easier.

In section 2 the nature of WWW links is described. Section 3 concentrates on the portability of complex HTML documents. The mirroring algorithm and the environment are presented in section 4.

	Caching	**Mirroring**
Time of downloading	On the first request	Before the first request
Downloading	Partial documents	Complete documents
Data persistence	Temporal	Persistent
Network load	Evenly spread	Burst-like
Allocated disk space	Dynamically changing	Static
Usage	Overall performance improvement	Fast access on a per document basis

II. Links in HTML documents

II.A. Standard notation of hypermedia links on the Internet

The language of World Wide Web documents is called Hypertext Markup Language [5]. Links in HTML files are expressed with Uniform Resource Locators [11] which give the information for a WWW client that is needed to retrieve the linked document. A URL has the following general syntax:

<protocol name>:<server descriptor>
<local descriptor>

where protocol name can be substituted with *ftp, gopher, news, telnet, mailto* or *http*. HTTP stands for Hypertext Transfer Protocol, the data transfer protocol of WWW [6]. Server descriptors consist standard IP machine names optionally accompanied by port numbers. Local descriptors provide the address for a part of the service inside the server, thus their format is specific to the protocol chosen. When it is meaningless, either the server or the local descriptor is omitted. Examples for the different types of URLs:

mailto:micsik@sztaki.hu
news:comp.sys.next.announcements
telnet://www.sztaki.hu:80
gopher://gopher.eunet.hu/00/ripe/About-ripe
ftp://ftp.sztaki.hu/pub/unix/INDEX
http://www.sztaki.hu/sztaki/contactinfo.html

II.B. Links between nodes of the World Wide Web

The http-URLs are discussed in more detail, since replicating other types of Internet services is not our intention yet. An http-URL can point at different types of data:

- hypertext file

  http://www.sztaki.hu/sztaki/contactinfo.html

- section of a hypertext file

  http://www.sztaki.hu/sztaki/contactinfo.html-phones

- in-line image

  http://www.sztaki.hu/pictures/logo.gif

- files of other types

  http://www.sztaki.hu/papers/TR95-1.ps

- executables

  http://www.sztaki.hu/cgi-bin/news.pl?comp.sys

The server response header contains the appropriate MIME type of the response content. This informs the client how to handle the content of the downloaded document. The client either presents it or activates the proper viewer application to display the content.

Generally a file path information is provided in the URL as a part of the local descriptor. In this case the information on a WWW server is stored on a per file basis inside the operating system's file system. The file path points to a file, the type of which can be deduced from the file name suffix. These suffices are mapped to appropriate MIME types which travel together with the content of the file to the client application.

II.C. Links to executable files

If an URL points at an executable file according to the configuration of the server, that file is executed, and the results are transferred to the client. This mechanism is done via the Common Gateway Interface [2]. CGI programs are recognized either by their suffix or by residing in CGI directories. Parameters can be passed to the script in two ways of which only that case is interesting for us, when parameters are encoded into the URL, just after the file path. Parameters are separated from the file path with a question mark. Even the file path may contain parametric information, the so-called extended path info, which starts right after the path of the program. For example:

http://www.sztaki.hu/cgi-bin/imagemap/sztaki/2ndfloor.map?12,12

this means that the cgi-bin/imagemap program is called with the extended path info of sztaki/2ndfloor.map and parameters 12,12. This CGI program handles clickable images. The parameter sztaki/2ndfloor.map points at the map file containing the

data of clickable areas on the image. The parameter 12,12 is the coordinates of the actual click.

According to present Web server software and their current usage, to decide whether a link is to an executable or to a solid file is not always possible. There are cases when neither any of the special suffices nor the parameter part are present, usually when the server has special directories containing CGI programs. There is no way to find out which are these directories on a server. The hidden danger for a mirroring program is to regard the output of a CGI program as a stationary file and thus creating a false mirror. Therefore it would be desirable to distinguish executables from other files by their suffices.

II.D. Absolute and relative links

Abbreviations of URLs are called relative links, where omitted parts of the URL are copied from the URL of the referencing document. In accordance to this an URL is qualified as absolute when all semantic parts of it are present. So let us consider this absolute URL:

http://www.sztaki.hu/sztaki/contactinfo.html

And have a look at some of its possible relative representations:

contactinfo.html (from the same directory)

/sztaki/contactinfo.html (from the same server)

//www.sztaki.hu/sztaki/contactinfo.html (with the same protocol)

In relative URLs one can also use the .. notation for the parent directory:

../../divisions/afe.html

An important property of URLs is that any URL can be converted from relative to absolute and vice versa regarding the URL of its referencing page.

III. Portable hypermedia documents

III.A. Structure of HTML documents

An HTML document could be considered as a set of files, and a set of links. The file set contains all files needed for the document. Files are generally HTML hypertexts but other file formats can occur as well. The link set can be split as real hypertext links, links for in-line images and links for executables or for other Internet protocols. The link set can also be split according the destination of the link, so outside links pointing out of the file set and inside links are distinguished.

III.A.1. Link structure

The link structure of an HTML document is represented by vertices as files and edges as links possibly labeled with a position inside the file. The most important feature of this directed graph is a way of connectedness, that in most cases all vertices are accessible via edges from a root vertex or call it an entry point. This is not a syntactical rule rather a semantic requirement otherwise the document can be considered as buggy or useless.

III.A.2. Storage structure

There is another secondary structure of an HTML document which we call storage structure. It represents how the file set is stored in the directory hierarchy of a file system. This structure is claimed to be secondary because this structure need not be revealed for the user and does not make an impact on the usage of the document.

Usually HTML format links are given with the help of the storage structure, so the link structure and the storage structure become closely related. However the storage structure of an HTML document can be modified while the link structure remains the same.

III.B. Idea of Portable Hypermedia

Generally moving of hypermedia documents consisting of several files corrupts the links inside them. Correction can be time-consuming (manual) work. A portable format for hypertext documents is needed. Documents in this format is freely movable inside file systems or between machines. Furthermore it can be served without changes from any WWW server or can be archived (compressed). This issue is spontaneously raised by some webmasters, and similar effort can be seen in Rohit Khare's *eText* software [7], but there isn't a well-established and standardized format yet.

A PHM is a directory or an archive containing the document file set, and a set of associated parameters. Parameters may contain: title, authors, copyright, abstract, entry point, file format descriptions, file format statistics etc. Minimally the entry point should be present. The included document file set has to meet the following requirements:

1. URLs pointing at files inside the file set are relative
2. URLs pointing outside the file set or referring to other Internet protocols are absolute

A set of different operations can be defined on PHM such as flattening the directory hierarchy of the file set or masking out some types of files (images, movies, sounds, etc.) from the file set. PHM format is to be used in our mirroring software environment for representing the result.

IV. A software environment for mirroring

A robust mirroring software was built with a clearly specified behavior and an extendible architec-

ture. First the two-phase mirroring algorithm is presented. After the architecture of the software environment is discussed.

IV.A. Two-phase mirroring algorithm

The mirroring algorithm navigates on the link structure of the HTML files which is a directed graph, so it is natural to use a specialized graph search as skeleton algorithm. The input of the procedure is an URL serving as an entry point of the result. The result is given in PHM format which will be moved to the desired final location.

The algorithm has two phases: first phase is downloading all related document files and in the second phase (relocation) the URLs are converted. The first phase operates with an URL set to be processed. This contains the URL of the entry point when the process is started. When a file is retrieved, links are extracted from it, analyzed and placed into the URL set. The next URL to be downloaded is chosen from the set, and that is the point where heuristics can be applied in determining the order of the retrievals. Retrievals are logged with enough information to handle time-outs, or fatal errors. In case of time-outs the retrieval is attempted again later. If the mirroring process is broken, it can be resumed at that point where the abort occurred, so earlier processing is not wasted.

The conversion of links takes place in the second phase after the retrieval of all files is finished and the URL set is empty. Then all URLs are analyzed again and converted to relative or absolute according the rules of PHM. This operation is called relocation of links. Postponing relocation after the retrieval phase has advantages: first it can be checked if the file pointed at by a local link exists, second it is easier to add several types of relocations or combine it with other operations on PHMs.

The skeleton of the algorithm:

```
program Mirror
(input: root-URL, options; output: result-PHM)
    Initialize URL set
    Check remote server's limitations for robots
    while URL set is not empty do
        Choose next URL for retrieval from the set
        (heuristics)
        Retrieve document for selected URL
        Handle errors, write log
        if document is successfully downloaded
        then
            if document has links then
                Select URLs to download,
                convert them to absolute,
                add them to URL set
            end if
            Store document
        end if
    end while
    for every retrieved document with links do
        Convert links to relative where necessary
    end for
    Create PHM as output
end program
```

IV.A.1. Classification of URLs with respect to retrieval

The program has to decide which files are to be downloaded and which files are not. The first limiting factor is a de facto standard on World Wide Web. Servers maintain a list of forbidden areas in their document space. This list is stored in a file called robots.txt at the root of the document space. Operations of automatic network retrieval processes (network robots) have to be limited compared to human users on Web servers, because robots can destroy the performance of the server by unintelligent repeating retrievals at high speed. For example robots can step into an infinite retrieval loop. This can be due to a bug or failing to recognize that they are retrieving documents generated by CGI programs. Falling into the robot category, our algorithm has to check the limitations for robots on the server.

Another limitation is that mirroring is done only via HTTP, all other types of URLs remain unprocessed. The functionality to handle multiple protocols can be added, but the result is questionable. One solution could be to integrate files retrieved via other protocols into the PHM, so from the mirror side they would be served via HTTP.

If an URL matches the above criteria, there are still two kinds of information which normally cannot be retrieved via HTTP: CGI programs and maps for clickable images. Therefore requests for these services are not mirrored but forwarded to the original host from the mirror site. With HTML 3.0 separate imagemap files will not be needed. CGI programs seem to be irrelevant to mirror, but still the safe identification of links to CGI programs is needed. A possible solution would be to standardize a set of suffices for CGI programs.

Also there are options which control the location and formats to be mirrored. A scope is given for determining the locations, URLs outside of the scope are not mirrored. Usually the scope is the directory of the entry point. Other scopes can be: the server of the entry point or several directories on a server. For the formats either wanted or unwanted file types are listed. This way one can mirror a document without images or without Postscript files, etc.

IV.A.2. Choosing the next URL to download

Document images should be downloaded right after the document itself. After the images a URL is chosen from the set applying heuristics. The set includes those links as well for which the retrieval failed with a time-out. The retrieval for these links

should be retired after well chosen intervals. Heuristics should take into account the actual behavior of network traffic and the user's expectations. The possible positive effects of using depth-first, width-first or more elaborate heuristic strategies under certain circumstances should be investigated.

IV.B. Architecture of the software environment

The environment provides services such as mirroring on-demand, timed mirroring, continuous mirroring. Mirroring on-demand can be initiated by any user. The environment collects the required document and sends it to the user in a PHM archive. Timed mirroring is a preprogrammed task where the document is downloaded at a preset time. This can help to perform mirroring in a low traffic period on the network. Continuous mirroring maintains stable copy of a changing document. This is achieved by intelligently repeated mirroring updates. During an update only the modified files are retrieved and put into the existing mirrored document.

The user interface for the environment is implemented as a set of WWW forms. The user can set the address (URL) of the requested document, the options for the mirroring software and the options for the result document format and the mode of delivery.

For continuously mirrored documents the preferred rate and time of the refreshment can be configured together with the previously mentioned options. A mirror scheduler is responsible for the control of the timed mirroring tasks. Every completed operation is logged, which is used when a broken mirroring is restarted. The mirror-log is stored and analyzed, if necessary, the next mirroring is rescheduled or the administrator is notified of the abnormal completion of the process. The following figure illustrates the elements of the environment.

IV.C. Experiences

Using the mirroring environment approximately 15 Internet sites have been mirrored, transferring 150 Megabytes of information.

One of the problems during the usage, was the lack of information about remote server's storage structure and conventions. It is embarrassing when the directory-index file (index.html by convention) is renamed and is referred with full name and with directory-referencing at the same time. Also, the server limitations for CGI programs mentioned in 4.1.1 (robots.txt) was missing several times.

The solution could be an improvement for HTTP protocol. For example, there could be a request to server like 'NNN Show server conventions'. The answer would be a text, each line containing a 'variable=value' pair, like 'directory-index=index.html' or 'binary-directory=/cgi-bin'. This should be cleared and a practical description should be made.

Another problem is the image-maps in HTML-2. Every image-map has a description which cannot be retrieved, so if the user clicks the map, a request is sent to the original site. The answer could be anything, e.g. HTML pages with links. These links could not be mirrored. In HTML-3 this problem is solved, the HTML-3 specification contains a well-defined manner of inserting an image-map into a HTML document, and in this way all points can be followed.

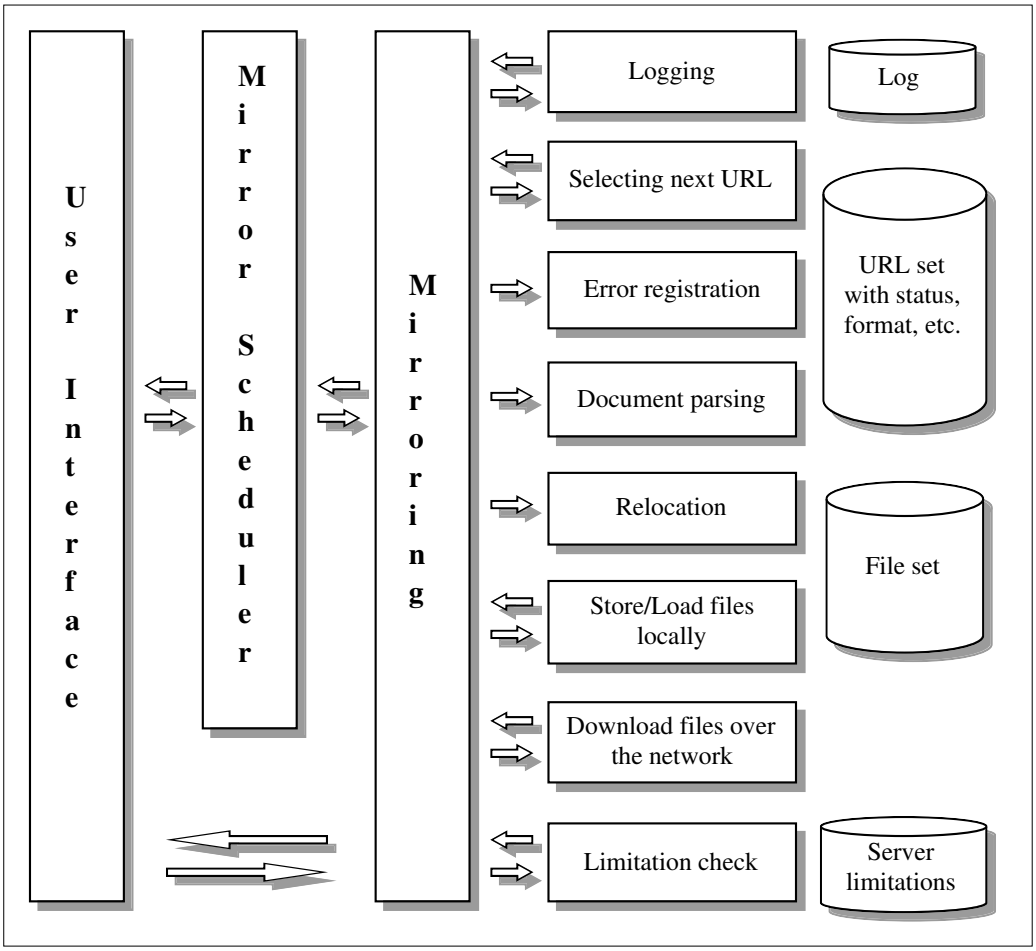
Perpetual usage of the program raised the problem of handling together downloaded Portable Hypermedia documents. If a document has a link to another document, and both are mirrored, but separately, the link in the local mirror will point to original place. One could expect that the link in the mirror points to the other mirrored document. To handle this problem, the idea of "mirrorspace" could be defined, which is a set of PHM-s, where links between PHM-s are converted to relative. Maintaining mirrorspace can be difficult. If one or more documents (text or image) are added to it, each existing document should be revised whether it have links to the added documents. Similarly, the newly added HTML document should be checked for every documents in other PHM-s. Deletion of one or more documents raises the same problem. These operations are very resource-demanding, takes a lot of time and data transfer. If one could solve these problems, the mirrorspace could be a good way of future development.

However, the program could be accelerated. Two passes of the algorithm could run in parallel. Parallel programming language implementation would be faster. Pass one (downloading the documents) could be divided by the URLs as each task retrieves an URL. There could be a limit for the number processes running at the same time, because this action burdens both the remote server and the network connection. Pass two (relocating) is an operation executed on pairs of elements of URL-sets, so it can be scheduled on URL-pairs.

V. Summary

A mature algorithm for mirroring and a standardized portable hypermedia format can ease the distribution of hypermedia documents through the World Wide Web. In this paper a two-phase mirroring algorithm was developed. The algorithm is able to create remote copy of a complex HTML document stored in another WWW server. The algorithm results the mirrored document in a portable hypermedia format (PHM) defined in this paper as well. Hypermedia documents in PHM format can be transferred with no need for further semantic transformations.

A software environment based on the previously mentioned algorithm for mirroring hypermedia documents was built. The environment provides different



high-level, intelligent automatic mirroring services via usual WWW interface (set of forms). The proper use of this software environment can decrease the network load during peak periods and can increase the accessibility of selected hypermedia documents.

The mirroring technique developed here can be the first step in the direction of introducing a separate protocol and/or protocol extension for mirroring purposes similar to that was proposed in [8].

VI. References

[1] Tim Berners-Lee, Robert Cailliau, Jean-Francois Groff, Bernard Pollermann: “World-Wide Web: the Information Universe”, Electronic Networking, Vol. 2. No. 1.

[2] Rob McCool: The CGI Specification, URL: http://hoohoo.ncsa.uiuc.edu/cgi/interface.html

[3] Douglas E. Comer: Internetworking with TCP/IP, Volume I., Prentice Hall International Editions, 1991

[4] Andrew Ford: Spinning the Web, How to provide Information on the Internet, International Thomson Publishing, 1995

[5] Hypertext Markup Language, URL: http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html

[6] Hypertext Transfer Protocol, URL: http://www.w3.org/hypertext/WWW/Protocols/Overview.html

[7] Rohit Khare: “eText: An Interactive Hypermedia Publishing Environment“ Proceedings of ACM Hypertext’93 - Demonstrations

[8] László Kovács, András Micsik: Replication within Distributed Digital Document Libraries. Proceedings of the 8th ERCIM Database Research Group Workshop on Database Issues and Infrastructure in Cooperative Information Systems, Trondheim, Norway, 1995

[9] L. McLoughlin et.al: Mirror - Mirror Packages on Remote Sites (UNIX man-pages)

[10] Oscar Nierstrasz, Gorm Haug Eriksen, Karl Guggisberg: w3mir, URL:ftp://sauce.uio.no/pub/src/w3mir

[11] WWW Names and Addresses, URIs, URLs, URNs, URL: http://www.w3.org/hypertext/WWW/Addressing/Addressing.html

Author Information

László Kovács works for the MTA SZTAKI, the Computer and Automation Institute of the Hungarian Academy of Sciences, as head of Distributed Systems Department. After his study he was involved in different projects in the areas of computer network protocol specifications, verifications and implementations. During his career he taught years in different foreign universities and research establishments including the University of Delaware, Newark/Delaware/USA and the Ecole Normale Supérieure de Cachan, Cachan/France. During the last years, his interests include research and development of distributed applications, World Wide Web services, CSCW, groupware systems, distributed digital library systems. At present, multimedia services, audio/video conferencing and virtual art are also included in his professional activities.

András Micsik works for the Distributed Systems Department of MTA SZTAKI. His activities include design and implementation of World Wide Web services, setting up audio and video conference environments, and teaching different Internet technologies at the application level. He is a Ph.D. student in Computer Science at Eötvös Loránd University, Budapest, where he got his M.Sc. degree in 1992. His research topics is about distributed digital libraries.

Gábor Schermann studies for his M.Sc. degree in Computer Science. at Eötvös Loránd University, Budapest. He also works for Distributed Systems Department, MTA SZTAKI. He is involved in implementing World Wide Web services as well as algorithms for digital libraries.