ALIWEB - Archie-Like Indexing in the WEB

By Martijn Koster (NEXOR Ltd)

version 0.1, March 16th, 1994

ABSTRACT

ALIWEB is a framework for automatic collection and processing of resource indices in the World Wide Web. The current ALIWEB implementation regularly retrieves index files from many servers in the Web and combines them into a single searchable database.

Using existing Web protocols and a simple index file format, server administrators can have descriptive and up-to-date information about their services incorporated into the ALIWEB database with little effort. As the indices are single files there is little overhead in the collection process and because the files are prepared for the purpose the resulting database is of high quality.

This paper discusses the background and objectives of ALIWEB and gives an overview of its functionality and implementation, comparing it to other existing resource directories in the Web. It reviews the experiences with the first months of operation, and suggests possible future directions of ALIWEB.

This paper is available in the Web on `<http://web.nexor.co.uk/mak/doc/aliweb-paper/paper.html>`

INTRODUCTION

The World Wide Web is a very exciting technology for deploying information on the Internet. In the last year the Web has grown both in terms of its use and the amount of published information.

As a side effect of this expansion it is becoming increasingly difficult to find things in the Web. This problem is known as *resource discovery* and occurs in any large information system. This paper investigates current methods for discovering resources in the Web, and presents ALIWEB, a framework to aid resource discovery in the Web.

OVERVIEW OF EXISTING METHODS

This section gives an overview of how the methods of resource discovery in the Web evolved, from the start of the World Wide Web project to the present situation.

Browsing

When there were only a few servers one could browse through the list of servers[1] maintained at CERN, and browse server home pages looking for up-to-date information. Now the number of servers prohibits this, it would take far to long to browse through all the home pages.

Listing

A number of people started to maintain lists of references to resources on the Web, such as the HCC NetServices list[2] and the NCSA Meta Index[3]. This meant users didn't have to visit all the server home pages, but could browse through an index. However, references became stale as documents and services moved or were revoked because these lists contained references to documents and services beyond the control of the maintainer of the index. Also, as new information became available this would only be added to the index if and when the maintainer found out about it and had the time to do so. Finally, as the number of resources on these lists grew it became cumbersome to look through them to locate a specific resource. The author himself maintained such an index, arranged by subject, but eventually gave up — The manual maintenance became too time consuming, and the index was not representative enough of the wealth of resources in the Web.

Searching

A solution to the problem that the index documents became too large to handle is to turn them into searchable databases, such as The GNA Meta Library[4]. This database still suffers from the problems of manual maintenance: it is time-consuming, and the information becomes out of date.

Automatic Collection

One of the latest searchable catalog is the CUI W3 catalog[5], which is based on automatic retrieval of a fixed set of documents that it has been specifically programmed to parse. As one of the documents it uses is the NCSA What's New list, this CUI W3 Catalog is very comprehensive and up-to-date. It doesn't solve all problems though, references still go out of date, there is duplication, and new sources have to be added by hand. In addition it has got a new problem: it is difficult to isolate information items with relevant context from some of source documents it uses, as they are not in a single defined format.

Automatic Discovery

There are indexers that take automation to the limit. It is possible to write programs that traverse ('walk') the World Wide Web, analysing and/or storing the documents encountered. These 'web-walkers' or 'spiders' have the advantage that they are very thorough, and can potentially visit all browseable parts of the Web. They are used for many different purposes; to find stale references, to discover new servers, to estimate the size of the Web, query databases for information, and they can index the Web (E.g. The JumpStation[6] and the RBSE's URL database[7], see the Robots Page[8] for others).

These systems too have problems. In the first place, spiders are potentially dangerous animals. As they retrieve many documents, they have considerable network overhead, especially when several robots operate simultaneously. They can request documents in such quick succession that the server becomes

[1] <http://info.cern.ch/hypertext/DataSources/WWW/Geographical.html>
[2] <http://www.eit.com/web/netservices.html>
[3] <http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/MetaIndex.html>
[4] <http://uu-nna.mit.edu:8001/uu-nna/meta-library/index.html
[5] <http://cui_www.unige.ch/w3catalog>
[6] <http://www.stir.ac.uk/jsbin/js>
[7] <http://rbse.jsc.nasa.gov/eichmann/urlsearch.html>
[8] <http://web.nexor.co.uk/mak/doc/robots/robots.html>

overloaded[9]. These problems regularly give rise to heated debates about robots. Currently there is an effort underway to prevent some of these problems: there are Guidelines for Robot Writers[10] and there is a Proposal for Robot Exclusion[11].

Another problem is the actual processing of documents. Because spiders come across plenty of documents it is very hard, if not impossible for them to get a sensible context for an information item. However the most important problem is that they retrieve all documents that can be reached, even those that aren't interesting or suitable to index. For example, the Hypertext Mac Archive[12] contains 5000 files, indexed into 106 folders, and allows searching for filenames and descriptions. If someone is looking for a Macintosh file, they can do so easily and quickly by browsing or searching from the Hypertext Mac Archive welcome screen. In contrast, a spider will not only retrieve the welcome screen, but all 106 folder indices as well. Not only does this mean the resulting database of documents is very big, it also means that any structure of documents is lost.

Parallels in other Information System

These problems of resource location in distributed environment are by no means new. Two other global information systems that have encountered this problem, and come up with solutions, are the Anonymous FTP sites, and the Gopher servers. The Anonymous FTP sites have Archie[13,] an automatic cataloging system that periodically retrieves listings of filenames from Anonymous FTP sites, via Anonymous FTP. These listings are combined into a searchable database, which can be accessed from the Internet via telnet, special clients, and gateways in other information systems including the Web14. Archie has been very popular, as it is very effective to find the kind of files people look for on FTP sites. Related files are usually combined in tar files, which gives a grouping, and placed in directories with descriptive names, which gives context to the search results

The Gopher servers have a similar system called Veronica[15,] which traverses Gopherspace and indexes Gopher menus, and provides a Gopher interface to search the resulting database. Although Veronica is also popular it is not always effective to find things, because there is no grouping of related Gopher menu items, so that you can easily get very many matches. There is also no structure information with the Gopher Menu items, which means there is little context to select items on.

Both these systems have proved extremely useful in their environments, and their success is in a large part due to the fact they operate automatically. For example in Archie, once an Anonymous FTP site has registered its desire to be included, no further bilateral effort is required to update the information.

All these approaches to resource discovery in the World Wide Web have their strengths and weaknesses. Even though all these approaches are useful, none is so universal or effective that there is no need for other solutions. This situation has prompted the design and implementation of ALIWEB.

THE ALIWEB FRAMEWORK

This section outlines the objectives, design, and implementation of ALIWEB.

[9] In a local test the number of documents a single spider requested from a lightly loaded server averaged 3 per second.
[10] <http://web.nexor.co.uk/mak/doc/robots/guidelines.html>
[11] <http://web.nexor.co.uk/mak/doc/robots/norobots.html>
[12] <http://web.nexor.co.uk/mac-archive/mac-archive.html>
[13] Deutsch, P. & Emtage, A., "The archie System: An Internet Electronic Directory Service," ConneXions, Volume6, No. 2, February 1992.
[14] <http://web.nexor.co.uk/archie.html>
[15] <gopher://gopher.unr.edu/00/veronica/veronica-faq>

Objectives

ALIWEB aims to combine the successful features of current resource location strategies in the Web, while minimising their disadvantages. Specifically, the objectives of ALIWEB are:

- To reduce the effort required to maintain an index.
- To reduce the effort required to search the index.
- To place low requirements on infrastructure.
- To help towards future systems.

What ALIWEB doesn't attempt to do is:

- Pretend to solve the Internet resource discovery problem.
- Make other searchable indices obsolete.
- Provide the only or best implementation.

Design

This section discusses the design of the ALIWEB model.

Decisions

To address the objectives a number of design decisions are of importance:

- To reduce the effort required to maintain an index.

The Browsing, Listing and Searching methods have shown that a manually maintained central database requires a lot effort, and is not practical on a large scale. It is preferable to index in a distributed manner.

To avoid extra efforts in development, learning, installing and maintaining new systems it is attractive to use the normal WWW protocols and mechanisms for this distributed indexing.

In a large distributed system scaleability needs to be considered, especially in the light of the recent growth of the Web. This implies open and extendible mechanisms.

- To reduce the effort required to search the index.

The Browsing and Listing methods have shown that letting the user wade through large amounts of indexed information is not very attractive. The searchable databases discussed in the Searching and Automated Collection methods ave proven to be more user friendly.

To prevent wasting effort following stale or double references it is important that the index is up-to-date and has no duplicate entries. This can be achieved by letting the information providers manage their own index, rather than relying on third parties.

To make the searching of a large index fruitful it is important to have a provision for extra meta information that can be used to narrow down search results, such as keywords and descriptions. It is also important that information providers themselves can decide what material should be indexed or not, to eliminate irrelevant information. Both these requirements make it easier to retain context and structure of resources.

- To place low requirements on infrastructure.

This requirement rules-out the use of Web traversing robots, as the overhead of retrieving or even checking every document on a Web server is very high both in terms of network traffic and server load, and unacceptable for many server administrators.

It is also another reason to use standard WWW methods rather then invent new ones that place an additional requirement on the infrastructure.

- To help towards future systems

As is the "way of the Internet" ALIWEB aims to provide a simple solution to a simple problem, not a solution that attempts to solve the entire resource discovery problem. Other systems exist and will come in place, and it is important to make sure that ALIWEB is as flexible as possible to satisfy future requirements, and that its information can be used by other systems.

It is to be hoped that the experience with the ALIWEB pilot can function as input in discussions about the resource discovery problem in the Web.

Overall Architecture

Summing up the design decisions we need a distributed indexing system where information providers have control over the indexing of their own information only, an automatic method of combining these index files, and a searchable database for searching the information.

This architecture of ALIWEB is very similar to that of Archie, hence the name Archie-Like Indexing in the WEB. One site retrieves index files from servers in the Web, combines them into a database, and allows the database to be searched (see Figure 1). However, there a number of differences, explained in section below.

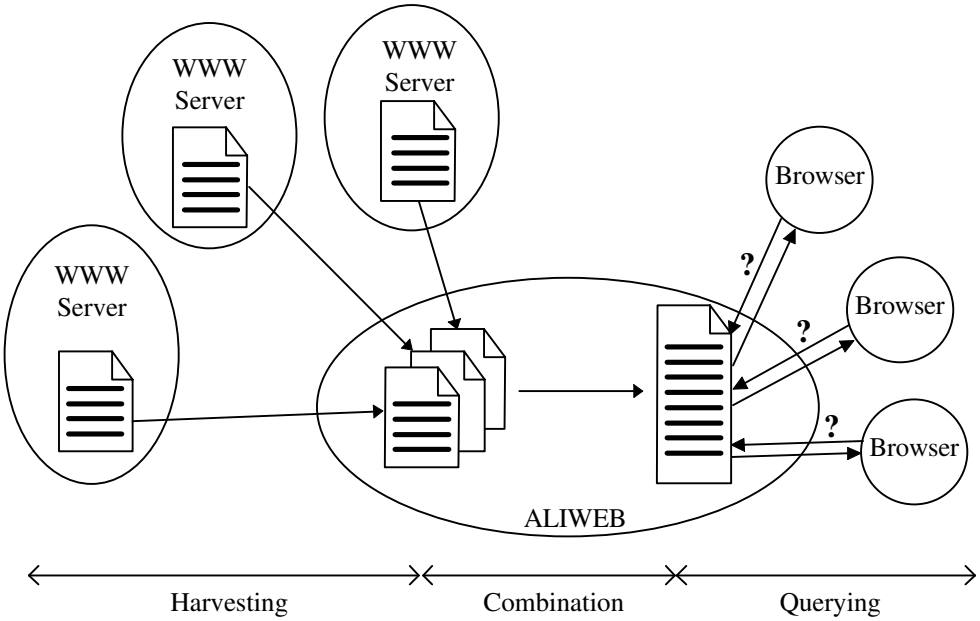

Figure 1: The ALIWEB Model

Index File Format

In Archie index files are produced by doing a recursive directory listing. This process provides basic information as filename, location and creation date, and can easily be automated. In the Web it is important to preserve context and structure, which is hard, if not impossible to automatically deduce from a document. So, ALIWEB uses index files specially prepared by the server maintainers. Currently most of these files are created by hand. This is less of a problem than it appears. Server maintainers will generally only include their main documents and services, not every single document on their

server. This preserves the structure of their services: people find their services with ALIWEB, and then use the structure of the services to satisfy their requirements.

These index files are text files in a specific format. Rather than invent a format to describe network services, ALIWEB uses the templates proposed by the Internet Anonymous FTP Archives (IAFA) Working Group[16] of the Internet Engineering Task Force (IETF). These IAFA templates were designed to contain information on networked resources found on Anonymous FTP sites, but can equally well be used to describe network resources provided by the World Wide Web. These templates are based on an attribute-value scheme, where attributes describe properties of an object. A number of objects can be described in a single file. The format in which these templates are stored is similar to RFC822 mail headers. For example, Figure 2 shows part of the index for the Web server at NEXOR Ltd:

```
Template-Type:     ORGANIZATION
Organization-Name: NEXOR
URI:               /nexor/nexor.html
Description:       NEXOR is an OSI software company.
Keywords:          NEXOR, OSI

Template-Type:     DOCUMENT
Title:             Perl
URI:               /perl/perl.html
Description:       Information on the Perl Programming Language. Includes a
                   local Hypertext Perl Manual, and the latest FAQ in
                   Hypertext.
 Keywords:         perl, perl-faq, language
```

Figure 2: Part of the /site.idx file of NEXOR Ltd's Web server.

Registration

Once a server administrator has created an index file, ALIWEB needs to be told about it. In Archie this registration is done manually. To minimise manual intervention ALIWEB uses a Web Form interface (See Figure 3) where people can register their index files interactively. This only needs to be done once.
By providing the domain name and port number of the Web server and the path of the index file ALIWEB has enough information to retrieve the file. In addition the name and email address of the person responsible for the Web server are required. These are used to contact the appropriate person in case of problems with the index file. Finally a frequency can be specified: as most index files are not likely to change frequently people can specify a minimum time-interval between retrievals to reduce overhead.
When the registration is processed ALIWEB retrieves the index file to ensure the details are correct, before accepting the registration.

[16] <http://src.doc.ic.ac.uk/public/computing/internet/ietf/iafa/iafa-charter.txt.Z>

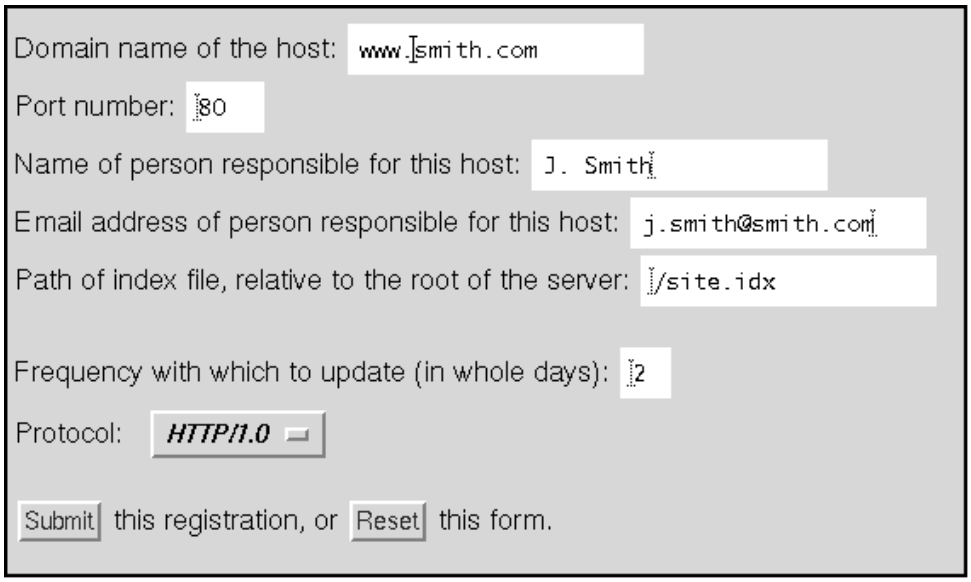

Figure 3: ALIWEB Registration Form

Harvesting

ALIWEB regularly retrieves the registered index files, parses the templates, and combines the valid entries into a database. This process is known as harvesting. In the current implementation the database consists of a single file with records similar to he ALIWEB index format, but the database could be implemented using different technology.

Searching

This database can be searched through a search engine. At the time of writing there are two ways to search the ALIWEB database: a simple search interface in the Web[17] (See Figure 4), or via other databases (e.g. the CUI W3 Catalog[18]).

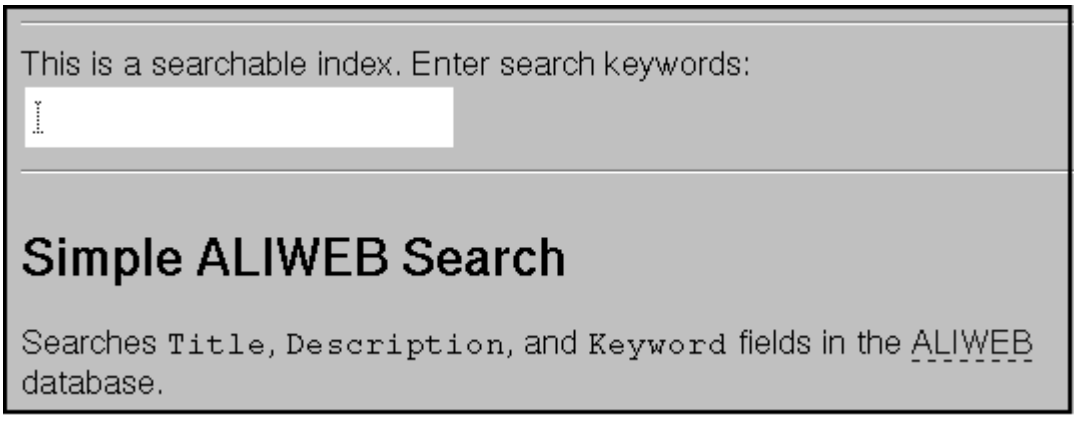

Figure 4: Simple ALIWEB Search interface

[17] <http://web.nexor.co.uk/aliwebsimple>
[18] <http://cui_www.unige.ch/w3catalog>

ALIWEB PILOT

To prove the validity of the model, and to provide a useful service a pilot service was implemented and deployed on NEXOR's Web server[19]. This section discusses aspects of this pilot.

Implementation

The current implementation of ALIWEB has been written as a collection of Perl[20]. scripts.
The harvesting process is a standalone process that retrieves the remote index files. It scans the list of registered users and compares the dates of the local copy of their indices to see if it needs to be updated. After a run it combines all the new and existing index files into a single database. This process is run automatically from UNIX cron.
The registration form and the Simple ALIWEB Search interface are modules running under the Plexus[21] Web server. The registration module directly updates the list of registered users, and the Simple ALIWEB Search interface searches the database for search terms.
A detailed description of the implementation is out of the scope of this paper, but further details, along with all program source and data files are accessible from the Web[22].

Operation

The pilot implementation of ALIWEB has been operational on web.nexor.co.uk since October 1993. During this time it has run on auto-pilot, with occasional manual intervention to resolve problems.

The number of registered hosts has grown to 54, with a total of 310 entries in the database. The Simple ALIWEB Search is queried an average of 80 times per day, and the database is used by the W3 Catalog in an average of over 8000 queries per week.

The main problem that became apparent during this period was that people registered their hosts without providing index files. This was caused by the fact that some people went straight to the registration form without really understanding what ALIWEB was. Two improvements were made to counter this problem:

- The documentation was rewritten and extended. This should make it easier for people to understand the concepts behind ALIWEB.
- The registration process was upgraded to include immediate validation of the presence and format of the index file being registered. Only valid registrations are accepted.

It still occurs that there are errors in the index files. Currently these are logged, and a manually run perl script mails the errors to the responsible persons.

There is another factor that has hindered the acceptance of ALIWEB (and systems like it) and which is increasingly becoming a problem: There is no standards committee or working group that provides a framework for defining protocols or operational procedures for the Web. Although this allows for a rapid evolution of the Web, it means that there is no channel to decide on standards for implementation and operation. This has implications for developers of browsers and servers, and indirectly for information providers. This lack of a framework for international coordination of development makes it difficult to deploy systems on a large scale.

[19] <http://web.nexor.co.uk/aliweb/doc/aliweb.html>
[20] <http://web.nexor.co.uk/perl/perl.html>
[21] <http://www.bsdi.com/server/doc/plexus.html>
[22] <http://web.nexor.co.uk/aliweb/doc/technical.html>

EVALUATION OF RESULTS

The pilot implementation demonstrates that ALIWEB meets its objectives.

The index is maintained with minimal effort, as the information providers only maintain indexes of their own information, no special protocols are required, and the registration, retrieval and combination processes are all automatic.

There is little effort in searching the index, a simple searchable document can be used. The framework allows for more elaborate search interfaces when they become required. The results of the search don't contain duplicate indexing efforts, and are as up-to-date as information providers make it. Similarly the information providers can add context information and retain the structure of his information by deciding what to include in the index and how to describe it.

The requirements on the infrastructure are low as only a single file is retrieved using normal protocols, and standard WWW methods can be used to prevent the retrieval of unchanged indices.

The ALIWEB pilot has generated some discussion on the comp.infosystems.www forum on USENET, on the www-talk mailing list, and on a dedicated ALIWEB discussion list, and is as such contributing to the discussion of the resource discovery problem. Its simple index file format makes it easy for other systems to use, and the CUI W3 Catalog has in fact incorporated ALIWEB into its database.

The main criticism of the current ALIWEB implementation is that some people think information providers are not willing or able to provide manually maintained index files. This is not in fact dictated by the ALIWEB implementation or the model — how the index file is generated is immaterial, as long as it is done in the required format. Automatic processes may well help with the maintenance of these index files. However, care must be taken to make sure the resulting index file has enough meta information to be selected on in a search.

The main criticism of the ALIWEB framework is that a single database for the entire Web is not realistic as it grows too big. Although the ALIWEB Framework can be extended to cope with this, as discussed in the next section, it is a valid concern. This problem depends on the granularity of the data, and it is for this reason that I suggest people limit their index file to small high-level descriptions of their services, rather than providing a full listing of every document on their servers. This touches the previous point; hand-maintained index information of high-level services is more useful than large automatically generated document listings without meta information. The decision of where the balance should lay rests with the information providers, who are in the best situation to appreciate the structure of the information.

FUTURE DIRECTIONS

ALIWEB has not originated from a planned development or as part of a funded project. As such the development is incidental, and requirement driven. Given that the current implementation meets the demands placed on it and is expected to do so for the near future no major development is required urgently. However, if the number of servers registering, or the number of documents indexed becomes very large, a number of possible developments are envisaged.

The index file format can be extended to include a categorical classification system. This could then be used by a forms-based search engine to narrow searches in the larger database, or use the categories to present virtual subject hierarchies. However, deciding on which catalogging system(s) to use needs further consideration.

If the number of registered servers becomes prohibitive for harvesting by a single site the harvesting can easily be distributed over multiple machines. Similarly the database can be mirrored by remote site so that search access can be distributed too.

Finally, if the combination of the multiple index files by a single site becomes the bottleneck, or the database becomes too large, several separate ALIWEB sites can maintain separate database devoted to a particular category or subject. It would be possible to build a hierarchy of ALIWEB servers, in the way WAIS has a directory of servers. Although the framework is flexible to this extent, the author believes that future directory systems with special purpose protocols will be used for advanced indexing purposes instead. It is expected that it will be possible for sites to convert their ALIWEB index files to new systems automatically.

CONCLUSION

Finding things in the World Wide Web is becoming increasingly difficult as the number of documents grows. Existing methods for resource discovery in the Web will not be able to handle this problem.

The ALIWEB framework provides a simple yet efficient method for distributed indexing in the Web. Experience with a pilot implementation has proven it to be a workable solution on a small scale, and there are indications that it will scale reasonably well, especially compared to existing methods. It has been shown that ALIWEB has some advantages over existing tools for resource discovery in the Web, while requiring a minimum of manual effort and networking resources.

There are criticisms of the ALIWEB Framework that it will not be effective as a single all encompassing database. This is not a purpose of ALIWEB, it is targeted as a high-level interim solution until other systems are developed to cope with the changed demands.

The author believes the ALIWEB implementation has a useful role in resource discovery in the current form of the Web. Although ALIWEB is not expected to become the final resource directory, the flexibility of its model gives it potential to become a major player in automated resource discovery in the Web in the medium term.