Supporting Resource Discovery Among Public Internet Archives Using a Spectrum of Information Quality[1]

Michael F. Schwartz
Darren R. Hardy
William K. Heinzman
Glenn C. Hirschowitz

CU-CS-487-90 September 1990

Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309-0430
(303) 492-7514

Abstract

Wide area networks offer access to an increasing number and variety of resources, such as documents, software, data, network services, and people. Yet, it is difficult to locate resources of interest, because of the scale and decentralized nature of the environment. We are interested in supporting a global confederation of loosely cooperating systems and users that share far more resources than can be completely organized. Therefore, mechanisms are needed to support incremental organization of the resources, based on the efforts of many geographically decentralized individuals, and a range of different information sources of varying degrees of quality. In this paper we describe a prototype implementation of a set of mechanisms intended to explore this problem in the specific domain of public Internet archives, accessible via the "anonymous" File Transfer Protocol. This is an interesting test case, because it encompasses a very large scale, administratively decentralized collection of resources, with considerable practical value. The resource discovery paradigm is exploratory in nature, with users contributing to the global resource space organization as they discover new resources. At present, three levels of information quality are supported. At the highest level, resources are described using an archive-site-resident database, with individual resources described according to their conceptual roles. Below that, per-user and per-user-site caches are maintained, to record resources that have been found by individual users during their explorations. At the lowest level, the system monitors announcements of public archive availability from USENET electronic bulletin board articles, to provide a simple keyword-based index of resources throughout the global network.

[1] †This material is based upon work supported in part by NSF cooperative agreement DCR-8420944, and by a grant from AT&T Bell Laboratories.

1. Introduction

The past five years have witnessed tremendous growth in the TCP/IP Internet, primarily through the highly successful deployment of the NSFNet backbone, and through growth in regional networks. Interconnecting academic, industrial, and government institutions, the Internet offers the potential for tremendous inter-organizational sharing of resources, such as documents, software, data, network services, and people. Yet, to date this sharing has been rather limited, consisting primarily of "arms-length" communication via electronic mail.[2] A relatively small amount of remote login and file transfer activity also takes place. However, because these forms of sharing usually require that permissions be arranged explicitly for each individual through human-administrative means, these more sophisticated sharing patterns are fairly restricted.

An interesting exception to these restrictions is the so-called "anonymous" File Transfer Protocol (FTP) [Postel & Reynolds 1985]. FTP is an Internet standard protocol that supports transfering files between interconnected hosts. Anonymous FTP is a convention for allowing Internet users to transfer files to and from machines on which they do not have accounts, for example to support distribution of public domain software. The primary means of securing systems that permit anonymous FTP is by limiting the domain of accessible file names, by setting the apparent file system name tree root of the logged-in FTP process to a limited subtree that does not contain sensitive files. Most anonymous FTP sites support read-only access to a set of files placed in a public directory by a system administrator. Some sites allow read-write access.

An enormous quantity and variety of resources are available through anonymous FTP. A single site can offer the user many megabytes of resources. Most resources are ASCII files representing software, technical reports, documentation, archived messages from interest group mailing lists, and databases. Binary-format executable files for specific common architectures are also available.

Unfortunately, it is difficult to locate resources of interest using anonymous FTP, because of the scale and decentralized nature of the environment. The available resources reside on tens of thousands of machines around the world, and there is neither a uniform convention for the organization of these resources, nor a global directory of the resources available through anonymous FTP. Given the decentralized administration of the Internet, this problem will become more pronounced with continued growth of the Internet.

In this paper we present an approach to this instance of the *resource discovery* problem, and discuss a prototype implementation of the approach. We do not address the problem of determining how to use resources once they have been located, although that problem falls within the purview of the larger Networked Resource Discovery Project, within which this work took place [Schwartz 1989].

The prototype we discuss is described in the context of a UNIX implementation, although there is nothing that keeps the tools from interoperating with other types of systems. Indeed, the underlying FTP protocol accommodates heterogeneity through the use of a standard set of commands and error return interface, and by automatically translating between a small number of common file data representations.

There are a range of reasons why people provide anonymous FTP archive sites, and these reasons have implications on resource discovery. Researchers use anonymous FTP to enhance rapid dissemination of their results, via software prototypes and technical papers. Corporations use anonymous FTP to provide software enhancements and general customer support. Network Information Centers and other institutions involved with network administration use anonymous FTP to improve the network infrastructure (e.g., providing software modifications to fix security problems, or network protocol enhancements to reduce network congestion). Interestingly, some sites (such as wsmr-simtel20.army.mil) do not directly benefit from providing anonymous FTP archives. Instead, individuals at those sites provide anonymous FTP out of a cooperative spirit. Given that the administrators of these sites are already performing a public service, any solution to organizing the accessi-

[2] Here we consider the number of users, rather than network traffic. File transfer currently generates the most network traffic [Horvath 1990].

ble resources must not require a great deal of effort on the part on site administrators.

1.1. Current Anonymous FTP Resource Discovery Paradigm

Currently, resources available via anonymous FTP are made known primarily through "postings" on various USENET electronic bulletin boards [Quarterman & Hoskins 1986], announcing the availability of information on a particular host. To gauge the volume of the announcements and scope of the public archive resource discovery problem, we scanned the contents of the news spool files at the University of Colorado on February 15, 1990, using a partially automated set of program scripts, plus manual browsing among the selected articles. This snapshot revealed that among 549 newsgroups received, there were 36,739 messages, consuming 96.5 megabytes of space. Within these messages were 214 different messages announcing the availability of some resource via anonymous FTP, in 89 different newsgroups. 98% of the news articles scanned were 2 months old or less.

These measurements indicate that even during a short period of time, a large number of resources were announced. Yet, because of the the high volume of news, relevant information is difficult to find. Moreover, because incoming news files are regularly deleted out of system spool directories as they age, these announcements are short-lived. Furthermore, because no standards exist for the description of archive sites, providing a high-quality directory of available resources is not easily amenable to automation.

In addition to these problems of scale, anonymous FTP users are also presented with a low level means to interact with archive sites. The typical paradigm involves logging in to a machine and performing a series of directory listing and directory change commands, in search of relevant resources. Searching for a specific file object can cause users to spend a large amount of time logging on and off remote machines and browsing through the directories. Because of this, many sites provide a top-level "README" file that describes pertinent information about the site's contents, organization, and access policies. Another common file is "ls-lR"[3] or "INDEX", containing a recursive listing of the files at that archive site, which the user may retrieve and browse locally.

1.2. Our Approach

Three premises guide the work reported in this paper. First, we believe that supporting sharing among a global confederation of loosely cooperating systems and users implies the existence of far more resources than can be completely organized. Therefore, mechanisms are needed to support incremental organization of the resources, based on the efforts of many geographically distributed individuals. Our second premise is that some resources are more valuable than others, and hence will merit more effort to organize. For this reason, a range of mechanisms are needed to organize resources and support resource discovery to varying degrees of quality, in accordance with the perceived value of the resources. Ideally, it will be possible to locate unpopular or relatively unimportant resources at greater expense, and resource importance will evolve over time. This assumption differs from the approach used in one of our earlier efforts, which biased resource discovery as a function of the size of a resource class, and its popularity within some neighborhood of users [Schwartz 1990]. Our third premise is that it is difficult to reach global agreement over information sources and formats, and hence that a number of different types of resource information sources should be accommodated.

Our approach involves an exploratory paradigm, whereby users initially try to locate high quality information about a resource, and progress to successively poorer quality information sources. Using poor quality information means that users will need to follow system-provided hints and apply human judgement to determine the best means for searching. During their exploration, users may choose to supply new information, to enhance resource discovery for resources of particular interest for future users.

---

[3] so-named because of the UNIX "ls -lR" command that performs a recursive directory listing in a long format that includes sizes, update times, etc.

While the general model involves an arbitrary number of different information sources of varying quality, our prototype supports three particular levels. At the lowest level, the system monitors announcements of public archive availability from USENET electronic bulletin board articles, to provide a simple keyword-based index of resources throughout the global network. The information at this level is collected using a set of heuristics intended to capture potentially useful hints for a broadly distributed range of sites. As users search through anonymous FTP sites, the system maintains two caches, to ease the task of finding one's way back to resources that were once visited. These caches represent the next higher level of information quality in our prototype. The first cache retains pointers to the directory names that were visited during searches. This cache is shared by all users at a site, and could be made available by anonymous FTP to users at other sites. The second cache allows users to maintain pointers to particular resources using explicit "mark" requests, for personal future reference. At the highest level of information quality, resources are catalogued using an archive-site-resident database, with individual resources described according to their conceptual roles. The tool that supports this level supports a manual means to generate the catalogue, as well as an automated (but less discerning) means to generate the catalogue, to reduce the human effort needed for this process.

Currently, the parts of the prototype are not integrated into a unified system, but instead exist as a set of tools to support the mechanisms we describe. We intend to integrate the tools in the near future.

The remainder of this paper is organized as follows. In Section 2 we discuss related work. In Section 3 we discuss the archive-site-resident conceptual databases. In Section 4 we discuss the per-user and per-user-site caches. In Section 5 we discuss the USENET article monitoring mechanism. In Section 6 we discuss some issues raised by our prototype implementation, concerning proper Internet protocol support for resource discovery. Finally, in Section 7 we offer our conclusions, and discuss future work.

2. Related Work

Several mechanisms currently address the resource discovery problem among Internet public archives. Announcement condensing services filter the contents of various bulletin board articles, summarizing or providing pointers to other source material. One example is the "comp.archives" USENET news group, maintained by a moderator who uses a set of program scripts to partially automate skimming through incoming articles, to find messages of particular interest to other users [Vielmetti 1990]. One particularly relevant message is posted on this news group monthly, describing the contents of various anonymous FTP sites. The list is provided by an individual who collects the information both personally and from the contributions of other people around the Internet [Granrose 1990]. As of this writing the list contained 665 different hosts in 408 different administrative domains, listing information about 1164 different topics. While significant, this list clearly only represents a small proportion of available resources, considering the measurements of Section 1.1.

"Clarinet" is a commercial service that provides newsgroups with structure like those of USENET, but with higher quality information (since it is controlled by a set of human editors), for a fee [Robinson 1990]. CompuServe provides a similar information service [CompuServe 1986], although Clarinet delivers its information to the user's machine, rather than requiring users to dial in to retrieve information.

Some sites (such as rodan.acs.syr.edu) maintain periodically updated listings of the directory names at various Internet archive sites, which can be retrieved by anonymous FTP and browsed locally by a text search program. The file name listings we observed encompass no more than a few dozen archive sites.

Other work on this problem includes archive sites that collect information related to specific interests, and post periodic announcements of the existence of the sites to various electronic bulletin boards. Examples include archives related to operating systems research (comp.os.research), telecommunications issues (comp.dcom.telecom), and MacIntosh software (comp.sources.mac).

While helpful, none of these mechanisms really solves the public archive resource discovery problem. Most of the mechanisms involve a good deal of centralized manual human effort, usually in the form of coordination centralized fashion by a human moderator. Such centralization slows progress and limits the perspective of the directory. Moreover, maintaining centralized directories does not scale well in the size of the network. Those mechanisms that do not involve manual human effort (such as the sites that retrieve directory listings of other archive sites) do not supply any conceptual meaning above and beyond the basic textual strings they provide. Most importantly, no attempt is made in any of these mechanisms to provide a general-purpose architecture in support of administratively decentralized resource sharing. In essence, while helpful, these mechanisms are only stop-gaps to a growing problem of the administratively decentralized organization of the Internet.

The more general problem of supporting shared information spaces has received considerable research attention. Schatz's Telesophy system uses hypertext technology to provide access to information shared by a community [Schatz & Caplinger 1989], albeit of considerably smaller scale than the collection of resources available globally through anonymous FTP. Whiteboards are an example of a still smaller shared information space [Donahue & Widom 1986]. A number of recent efforts have been devoted to the problems of supporting access to a digital library systems, intended to provide access to a range of documents [Arms 1989, Kahn & Cerf 1988].

3. Information Quality Level 1: Archive-Site-Resident Databases

The top level of information quality is oriented towards sites that are willing to put a reasonable level of effort into organizing their contents. The tool we have built that supports this level of information quality is called *aftptool*. *Aftptool* plays a dual role, providing the site administrator a means of documenting resources, and supporting users in discovering resources.

3.1. Site Administrator Support for Organizing Resources

To assist the site administrator in organizing the files in an archive, *aftptool* provides a database to maintain high-level conceptual descriptions of the resources, and a keyword-based index of the resources. Because the information is stored in a database, more complex queries can be issued than are supported by the typical "ls-lR" and "README" files found at many anonymous FTP sites. Descriptions may be associated with individual files or with groups of files. For example, many resources are application source code. In this case, a logical grouping within a subdirectory is a realistic request, presumably users will want to retrieve all of the source files. The archive site administrator then only needs to document the subdirectory, rather than each individual file.

Aftptool uses "gdbm", the GNU database package, to hold archive site databases. An example of the contents of one such database is shown in Figure 1. The fields in each record contain file names, path names, the FTP command needed to cause the file transfer, a description of the resource, and keywords describing the resource.

```
%A /pub/X/contrib
%B andrew
%C get andrew
%D Andrew -- X windows interface prototyping tool, from Carnegie Mellon University.
%K user interface source code cmu tool prototype X windows browse
```

Figure 1: Example *Aftptool* Site Archive Database Contents

Although *aftptool* allows a site administrator to enter information into the pertinent fields in the site archive database, doing so can be labor intensive. Therefore, an automatic mode of database creation is also provided. In this mode, *aftptool* generates a default set of keys based on the path names in the archive, excluding the leaf nodes of each path name. For example, the file "pub/os/sun/tools/dd1.c" would generate the keys "pub", "os", "sun", and "tools". The leaf path name nodes are not used because these names usually do not impart much additional meaning, beyond the context of their full paths. The description generated in automatic mode consists of the final directory name of the data ("tools" in the above example).

Clearly, automatic mode generates less meaningful database entries for the resources. It cannot generate truly conceptual descriptions of the resources. Moreover, automatic mode can generate a large number of keys (often ten times as many as would be entered manually), , and can miss some relevant keys (as in the case, for example, where a software distribution is "bundled" into a single "tar file", whose name is excluded as described above). Another problem with automatic key generation is that no distinction is made between logical groupings of files in a subdirectory vs. individual files, since there is no easy way to decide automatically whether a set of files in a directory are related, or just happen to reside in the same directory.

Because of these problems, the most appropriate use for automatic mode is to generate a basic set of information about a set of resources, which is then modified and improved by the site administrator. The site administrator can manually prune through and add keys after they are generated, and can add conceptual descriptions at this time as well.

Regardless of whether the site administrator is building the database automatically or interactively, the first step is to build a temporary file containing the results of a depth first recursive listing of the all the subdirectories and their files, generated using the UNIX "ls -lR" command. This listing is then parsed into subdirectories and their files. If *aftptool* is being run in automatic mode, the keys for every file in the subdirectory are built from the path names as described above, and the title field is generated using the file name. If the data base is being built in interactive mode, the site administrator is prompted for a title, short conceptual description, and key words, either for each file individually, or for the subdirectory as a whole. Based on the response, *aftptool* adds the FTP command to retrieve individual files or all the files in the subdirectory. For example, a subdirectory that contains the source and include files for an application will be documented as a whole with the appropriate FTP command to transfer all of the files (mget *). For a subdirectory containing RFCs, each file would be documented individually, again with the appropriate FTP command (get "filename"). An example site administration session is illustrated in Figure 2.

```
Build Database Automatically (y/n)? n
Directory pub/X.V11R4/FIXES/contrib/andrew/ has 8 files.
 1  patch.001.Z
 2  patch.002.Z
 3  patch.003.Z
 4  patch.004.Z
 5  patch.005.Z
 6  patch.06a.Z
 7  patch.06b.Z
 8  vui.patch.Z

Catalog separately or together (s) or (t)? t

Enter Title: Patches for CMU's andrew interface tool, for X11R4.
Enter key words separated by spaces, maximum of 255 characters.
> X X11R4 andrew source cmu UI Carnegie Mellon University
```

Figure 2: Example of Anonymous FTP Site Administration Session

Comparing this representation of archive site contents to the structure of an anonymous FTP site directory without the database indicates how *aftptool* improves resource discovery. FTP treats files as unstructured byte streams, without any semantics. While descriptive file and path names help to some extent, the user of anonymous FTP must first search through a hierarchical arrangement of these files before any meaning can be discerned from a particular name. In contrast, *aftptool* allows humans to associate meaningful keywords with a resource, so that it may be found without exploring a hierarchical file name tree. Moreover, *aftptool* allows one to associate a conceptual description with each resource, so that its semantics may be discerned without first retrieving and examining it.

3.2. User Support for Discovering Resources

Aftptool presents the user with a quick and easy way to discover the resources of an archive site. The user is first prompted for the remote site name (*aftptool* does not provide support for locating known sites; that support is provided by other parts of the prototype). *Aftptool* connects to the remote site, logs in as user "anonymous", and retrieves the archive database. The user then uses *aftptool* to search the database locally, using keyword queries. The user can also view all of the keywords in the database as a means of finding resources. This capability is particularly useful if the database was built in the automated fashion described in Section 3.1.

After discovering an interesting resource, the user can choose the the resource to retrieve. *Aftptool* then uses the path name, file name, and command in the archive site database to retrieve the resource. After each retrieval, *aftptool* returns to the root of the anonymous FTP site, ready to retrieve the next resource(s). From the user's point of view, the location of the file and the details of navigating through the archive site's name tree are transparent. An example user search session is given in Figure 3.

```
Enter host to which you wish to connect: latour.colorado.edu
Do you want to browse (b) or keyword search (k)? k
Enter key: Xwindows

8 resources matched.
1  pub/X.V11R4/FIXES/contrib/XView/README
2  pub/X.V11R4/FIXES/contrib/Xcu/
3  pub/X.V11R4/FIXES/contrib/Xw/Xw.fix01
4  pub/X.V11R4/FIXES/contrib/gwm/
5  pub/X.V11R4/FIXES/contrib/kinput/
6  pub/X.V11R4/FIXES/contrib/winterp/patch-0
7  pub/X.V11R4/FIXES/contrib/xplaces/xplaces.fix03
8  pub/X.V11R4/FIXES/mit/

Choose an item for description (c), or quit keyword search mode (q)? c
Enter number of item: 6
The title of the resource is "Patches for winterp for version R4".
Retrieve this resource (r) or continue (c)? r
Resource will be placed in the directory /piper/student/heinzman
```

Figure 3: Example Resource Discovery Session Using *Aftptool*

3.3. Discussion

Aftptool addresses three basic resource discovery problems. First, it enriches the organizational structure of the archive site, by using a database to describe resources held there, and by encouraging site administrators to construct conceptual descriptions of the resources at their sites. Second, because building a database describing the resources can be labor intensive, *aftptool* provides support for automating the process. Finally, *aftptool* reduces the burden of retrieving files, by hiding the mechanical details of this process. The user simply requests a resource (which may involve many files on a remote machine), and *aftptool* automatically connects to the remote machine, moves to the appropriate directory, and retrieves the files.

There are several benefits of using a database for organizing resources available through anonymous FTP. First, the database can support more sophisticated operations than are typically available when searching the "ls-lR" files, such as searches on logical combinations of keywords. Second, once the database has been retrieved, the user will perceive little network latency. Given that future networks will offer significantly increased bandwidth but only moderately reduced latency, retrieving the database and searching it locally is worthwhile. For networks links that have relatively low bandwidth, a future resource discovery protocol could support running the database access program at the remote site. Doing so would require less bandwidth than the typical means of searching for files now, using a series of "ls" and "cd" commands.

While *aftptool* eases the problem of resource discovery at a particular site, it does not address global resource discovery over many anonymous FTP sites. A clear future improvement will be to incorporate the ability to merge multiple archive site databases into a database at a user's own site, or at a well-known anonymous FTP site.

The source code for *aftptool* consists of 840 lines of C. In addition, some minor modifications were made to the FTP client code, which is nearly 7000 lines long. Gdbm adds another 4200 lines of C code.

4. Information Quality Level 2: Per-User and Per-User-Site Caches

The goal of the second level of information quality is to build a guide to the resources available via anonymous FTP as users explore these resources. The hope is that if a sufficient number of people participate in network exploration, a fairly complete "map" of the archive sites can be quickly compiled. Furthermore, since many anonymous FTP sites will not assist in cataloging their resources, there must be local support for resource discovery. A good way to fulfill these requirements is to cache information as it is discovered, during the course of the usual anonymous FTP searching paradigm of repeated directory listing and directory change commands. As usual with caching, this technique will work well if there is some locality of access to the data. In the present case, caching information about public archive contents will work well to the extent that users that share a cache have similar resource interests. The exact nature of the caches we support will be outlined shortly.

To implement this idea, we modified the standard FTP client program to support caching. We call this program *aftpcache*. To make the cached information most useful, *aftpcache* divides the information into categories through the use of keywords. For example, if users on a system are interested in UNIX source code and UNIX applications, they can divide their cache into two categories. *Aftpcache* uses these categories to divide cached data, showing the host names and subdirectories where relevant located information has been found:

```
UNIX-src
    /pub/UNIX/device_drivers@machineA.domain.name
    /pub/source/UNIX/berkeley_4.2@machineB.domain.name

UNIX-app
    /UNIX/applications/text_editors@machineC.domain.name
    /pub/code/UNIX/games@machineD.domain.name
```

As was done in the top level of information quality (*aftptool*), only directory names are cached, rather than leaf file names.

In our initial approach, *aftpcache* retrieved a recursive directory listing of each anonymous FTP site probed by the user. Directory listings were retrieved after the user had disconnected from all sites, to minimize noticeable effects of bandwidth limitations during the interactive session. After retrieving this information, this version of *aftpcache* would extract the directory names from the listing, and cache directory names whose final components exactly matched a pre-selected list of category names. An aliasing facility was also provided, to group related resources under a common category title. For example, the UNIX source code category might have looked like:

```
UNIX/UNIX-src/srcs/source/sources
    /pub/UNIX@machineA.domain.name
    /pub/general/code/UNIX@machineB.domain.name
    /bin/op_systems/src@machineC.domain.name
```

The drawbacks to this method became clear fairly quickly. If an anonymous FTP directory contained resources pertaining to a particular category but its name did not exactly match any of the aliases, it was not entered into the cache. We tried defining a "default" cache category to hold directory names that did not fit anywhere else in the cache, but this category filled up quickly, containing so much unstructured information that it was of little use. Moreover, the amount of information retrieved by recursive directory listings could be unacceptably large, often exceeding a megabyte per archive site. Leaf file names account for much of the information retrieved in this manner, yet we do not cache leaf file names anyway. Hence, this method was deemed too wasteful of network bandwidth and cache space on the local disk.

At this point, we came to the conclusion that *aftpcache* was trying to satisfy two different goals. The first goal is to assist users in discovering what resources are available. The second goal is to assist users in *resource management*, or keeping track of interesting resources that they have previously found (a problem some refer to as *information retrieval* [Fischer & Stevens 1990]). Essentially, the problem with our initial caching mechanism was that it tried to address both goals with a single cache. Therefore, we modified *aftpcache* to use two separate caches, with two separate cache management policies. Furthermore, we modified *aftpcache* so that it did not retrieve and cache entire recursive directory listings of probed sites.

4.1. Resource Discovery Cache

The resource discovery cache is shared by all users at a particular site (such as a department or a company). Users of *aftpcache* therefore get the benefits of seeing the cached explorations of previous users, and their explorations are recorded for future users. Resources are organized by their directory names as before, but now the category names simply reflect the machine on which the directories can be found.

The caching policy used for the resource discovery cache is to save information in response to the movement of the user through the remote site. In other words, the only directories that are cached are those into which the user explicitly requests *aftpcache* to move. The motivation for this change is the assumption that users will only visit directories that contain resources of interest to them, and that these directories will likely be of interest to future users at their site. This change drastically reduces the storage requirements of *aftpcache* over caching entire recursive directory listings.

The possibility exists, of course, that something very useful to another user might get skipped over by this policy. Yet, we prefer missing some resources over having *aftpcache* guess what is valuable or, worse yet, retrieve all of the available information and make the user sift through it. Thus, even if a user locates nothing on a remote site that is of interest to someone else, a second user at least has the machine's name and can can chose to investigate that site. Moreover, the discovered information accumulates over time, so that if the second user moves to a different part of this archive site's name tree, the discovered directories will be added to those found by the first user, thereby offering future users the sum of their explorations.

A second reason why the cache size remains reasonable is that the data storage technique is more intelligent than that used by the original prototype. Originally, a directory was stored by its full path name. Thus, if *aftpcache* added two sibling subdirectories, the cache might contain something like:

```
UNIX-src/UNIX-app
        /pub/code/UNIX-src@machineA.domain.name
        /pub/code/UNIX-app@machineA.domain.name
```

In this case, the directory names and the archive site names are repeated. With the new mechanism, the full resource name is stored only once. *Aftpcache* maintains the full tree structure by storing a directory tree's level and relative tree location along with its name. Thus this example becomes:

```
machineA.domain.name
        / F1pub F2code F3UNIX-src N3UNIX-app

pub is the (F) first child of / and it is on level 1.
code is the (F) first child of pub and it is on level 2.
UNIX-src is the (F) first child of code and it is on level 3.
UNIX-app is the (N) next sibling of UNIX-src and it is on level 3.
```

For this small example, a 42% reduction in space (62 vs. 36 bytes) was realized, excluding category names. Of course, this encoding is transparent to the user.

4.2. Resource Management Cache

A separate resource management cache is maintained for each user on the user's local host. As with the first prototype, the unit of storage is the directory name, with the anonymous FTP site attached. However, with the resource management cache, users must explicitly declare that they wish to save a pointer to the directory, using a simple *mark* primitive. When a user requests that a directory be marked, *aftpcache* prompts for a category under which to store it in the resource management cache. If the category already exists, the directory is appended to its directory list. Otherwise, a new category is created, and the directory name is entered as the first member. Furthermore, we modified the matching algorithm so that category names no longer need to match the resource names within them. Category names can then reflect logical organization.

To allow the user to better manage this cache, *aftpcache* has a category manager that allows users to add, delete, and merge categories. Using this cache manager, users may adapt the cache to suit their changing notions of how resources are related to one another throughout the Internet, independent of how directories are named across Internet sites.

4.3. Example Session

Figure 4 shows an example session of using *aftpcache.* When first issuing the browse command, the user is placed at the top level of the resource management cache, and the categories are listed. In this particular cache, there are only two categories: *graphics* and *example*. The user chooses the *example* category, and *aftpcache* then lists the member directories of this category. There is only one member directory, which the user selects. *Aftpcache* then logs the user into the hosts that holds that resource. The user then disconnects and begins to browse the resource discovery cache, which holds information that has been explored on particular hosts. There are only two hosts listed. The user then requests that *aftpcache* display the directories contained for the second anonymous FTP site.

```
ftp> browse
0: graphics
1: example

(D)isplay ALIASES (N)ext screen (Q)uit (S)can HOST CACHE or number of ALIAS? 1

0: /cisco/test@spot.colorado.edu

(D)isplay ALIASES (N)ext screen (Q)uit (S)can HOST CACHE or number of ALIAS? 0

Connected to spot.colorado.edu.
220 spot.colorado.edu FTP server (Version 4.1 Sun Aug 7 19:42:25 EDT 1988) ready.
331 Guest login ok, send ident as password.
230 Guest login ok, access restrictions apply.
250 CWD command successful.

browse> pwd
257 "/cisco/test" is current directory.

browse> close
221 Goodbye.

0: graphics
1: example

(D)isplay ALIASES (N)ext screen (Q)uit (S)can HOST CACHE or number of ALIAS? s

0: boulder.colorado.edu
1: spot.colorado.edu

(D)isplay HOSTS (N)ext screen (Q)uit or number of HOST? 0

/
    3b1
    TeX
    X-seminars
        X-toolkit
        Xaw
    usenet

(D)isplay HOSTS (N)ext screen (Q)uit or number of HOST? q
```

Figure 4: Example *Aftpcache* **Session**

Aftpcache involved modifying or adding approximately 400 lines of C code to the standard FTP client, as well as interfacing this code with gdbm.

5. Information Quality Level 3: Monitored USENET Articles

The third level of information quality is oriented towards providing a simple means of finding even vaguely relevant information about the broadest possible range of topics spanning as many archive sites as possible. We have built a tool called *aftpgather* for this purpose. By using a number of simple heuristics to scan the unstructured information available in USENET articles, *aftpgather* generates a simple keyword-based index of potential public archive sites. With this index users can then form conceptual ideas about the global network, and make educated guesses as to which potential public archive might have the desired software. Regular expression matching facilitates searches of this index, leading to a quick search of the global network based on little initial information.

Aftpgather scans newly arrived USENET articles from a selected set of promising newsgroups (such as comp.archives and comp.sources.wanted) once per week, looking for keywords such as "anonymous," "ftp," and "archive". Generally, when people announce the availability of software, only one site or software package is discussed. Therefore, for each article containing a possible archive site announcement, aftptool builds an index that associates the keywords found in the article with the hostnames and Internet addresses found there. This association allows any of the keywords to retrieve the host names and articles at lookup time. The list of keywords, hostnames, IP addresses, and words found in lines containing keywords are saved as an additional information section of each resource record. This information is helpful when searching the keyword-based index, as it helps the user form ideas about not only what these potential public archive sites have to offer, but also other hosts possibly holding related resources.

In generating the index, *aftpgather* applies a number of simple heuristics to help exclude irrelevant keywords, that we devised through iterative experience. First, long articles are excluded from consideration, because they tend to generate too many keywords, degrading the quality of the individual keywords. Second, if a line has too many words or too few long words, it is omitted from the index, because lines containing a few long words more often contain useful keywords. Third, words less than four characters long are excluded, since short words often make poor keywords. Finally, if a word is found to be in a list of common words (such as "the", "and", etc.), it is excluded.

Because the index associates all of the keywords extracted from an article with all of the hostnames and Internet addresses found, an article that has a large number of unrelated keywords will degrade the quality of the data because of the "crosstalk" between the keywords. Ironically, one place where this problem occurs is in articles listing public archive sites, such as the anonymous FTP site list discussed in Section 2. To reduce this problem, articles are saved and marked for later reference when they contain too many hostnames or IP addresses. The site administrator or the user can look at the full article text to clarify the hints from the displayed information.

The database collected by *aftpgather* can be searched either by host (using *aftphost*) or by keyword (using *aftpkey*). Figure 5 shows an example using each of these tools. First, a host name is given as a keyword. This search results in showing the hints about the available software at that host. In the example, the database shows that boulder.colorado.edu might have a resource called "mactivation". Furthermore, a human might infer that this resource is software related to neural networks. Next, the user uses a keyword to find which host might software related to that keyword. In this example, the user might infer that decwrl.dec.com has some technical reports on virtual memory systems.

We considered making *aftpgather* retrieve directory listings from the sites it finds. We decided against doing this because we considered it to be too expensive, and unlikely to yield much useful information. It seems more reasonable to let users guide the searches, and save the discoveries in the per-user and per-user-site caches.

Aftpgather is implemented in C and PERL (the Practical Extraction and Report Language, an interpreted language that supports many of the features of a full programming language and system call library [Wall 1989]), and used the GNU dbm library. The code totals about 2000 lines, two thirds of which were C, and one third of which were PERL. PERL was effective in retrieving the USENET articles from the NNTP server and quickly parsing them. Since PERL only supports standard UNIX dbm, C was used to interface with the GNU dbm routines. After scanning approximately 1,000 articles, the database is approximately eight megabytes long. Most of the size of this database is caused by gdbm's data representation, plus some smaller inefficiencies in the way our code stores data in the gdbm records. For example, currently an 11 megabyte gdbm file contains less than 250,000 bytes of "real" information. Clearly, this is an area we will need to improve prototype in the near future.

The use of heuristics for scanning USENET articles was inspired by our earlier work on a "white pages" directory tool that locates electronic mail and postal address information about Internet users [Schwartz & Tsirigotis 1990]. We call this technique *exploiting application semantics*, because it involves building an understanding of the semantics of a particular application into the algorithms that support searches. The advantage of this technique is that it permits resource discovery given simply structured information. This is essential in heterogeneous, administratively decentralized environments, where global agreement is difficult to reach over highly structured information formats. Our experience with the Internet white pages tool indicates that this technique is a powerful way to support resource discovery. Indeed, that tool is capable of locating information about over 1.1 million users distributed across approximately 1,900 administrative domains around the Internet. Moreover, it has been our experience that the process of evolving the heuristics used by the technique of exploiting applica-

```
eclipse% aftphost boulder.colorado.edu
    Host:   boulder.colorado.edu
    Keys:   128.138.240.1, author, called, info, kranzdorf, mactivation, mactivation/27-aug-90, mike, net,
            network, neural, original-subject, simulator, version
    Additional information:
            128.138.240.1, anonymous, archive-name, archive-site, author, available, boulder.colorado.edu,
            called, from, info, kranzdorf, mactivation, mactivation/27-aug-90, mike, net, network, neural,
            original-subject, simulator, version
    Full Articles:
            /latour/users/hardy/saved_articles/{327,342}

eclipse% aftpkey virtual
    Key:
        virtual
    Hosts:
        decwrl.dec.com
    Additional information:
        90/4, archive-name, archive-site, authors, available, file, help, line, memory, nelson, original-subject,
        publication, reports, research, subject, system, title, virtual, western, with, word, wrl-techreports/27-
        aug-90, wrl-techreports@decwrl.dec.com
    Full Articles:
        /latour/users/hardy/saved_articles/{73,121,204,212}
```

Figure 5: Example Usage of USENET Hint Lookup Tools

tion semantics leads to further insights about resource discovery.

6. Discussion

While the prototype tools we have built help with the resource discovery problem, they also underscore ways in which a more powerful sharing paradigm could be achieved, given a more suitable protocol than anonymous FTP. In this section we discuss some of the conceptual shortcomings of anonymous FTP, as regards global sharing.

A major problem using anonymous FTP for supporting global sharing is the fact that FTP was designed with a point-to-point paradigm in mind. The protocol has no notion of distribution, and no means to provide linkages between sites. One can store files in public directories listing linkages (e.g., listing other relevant sites to check), yet this type of linkage is not part of the protocol, and can only be used by users or programs separate from FTP. If FTP did support such inter-site linkages, it would be possible to build a distributed data structure linking related sites, or providing some type of global indexing/directory scheme. Instead, intersite organization takes place "outside the system", primarily in news articles. Besides the obvious problem of making it difficult to find existing resources, the lack of intersite linkage and directory support within the protocol means that users may waste a good deal of network bandwidth, repeatedly connecting to sites, performing directory listings, fetching inappropriate files, etc.

A related problem is the fact that FTP incorporates a very limited notion of trust, by exporting underlying file system authorization mechanisms that were designed for centralized systems to users from outside administrative domains. For example, the UNIX file system protection mechanism only allows one to define access permissions for file owners, group members, and all others. There is no distinction between outside administrative domains. Therefore, it is not possible to allow some sites to have update access to an anonymous FTP archive without giving that access to all Internet users, or giving regular accounts to some subset of users. This fact makes it difficult to use the disk space provided by anonymous FTP sites for storing intersite linkages. Alternatively, the site administrator could act as a central coordinator of organizational information. However, doing so unduly centralizes the effort and authority of the distributed directory.

Also related to the problem of forming intersite linkages is the fact that FTP does not provide a general means to run programs on behalf of clients. The only software that a client may run using FTP are various built-in commands (such as changing directories and retrieving files), and one specific program for performing a directory listing ("ls" in the case of UNIX). If the protocol were just slightly more general, it would permit site administrators to place any program in a specified directory containing executable programs, rather than just ls. This would, for example, allow an implementation of *aftptool* to run the database lookup software remotely, rather than first retrieving the index across the network. For network links with low bandwidth and reasonably low latency, this may be worthwhile.

Another problem with using FTP for supporting global sharing is that there are no common formats for file naming and data representation. As a result, each site currently decides its own formats. Sometimes these formats are described in "README" files. In this case, however, it would be difficult to make an automated means of inferring the organization; a human must apply conceptual reasoning abilities to understand the organization. Even in such a case, the organization of a large archive site may be so complex or counter-intuitive that even experienced humans cannot easily find the resources at that site. This is the case, for example, at several sites we have seen that group files by physical disk location rather than by logical organization.

We do not advocate requiring a global standard in this regard. We believe it is too difficult to reach consensus on such standards. Even if standards evolved, there would still be the problem of upgrading existing information to fit the standards. Instead, we believe it would be worthwhile to support a number of formats, and to use some type of language to describe the particular formats being used, so that a program may connect to a site, determine the format(s) in use, and properly interpret the organization. We will discuss this notion further in Section 7.

Another problem is the fact that FTP's primary abstraction is that of a file containing an uninterpreted stream of bytes. Even given support for specifying global organization and data representation, FTP does not associate any notion of semantics with files. While a file typing system would help this problem, we believe that typing systems are typically oriented more towards machine automation than towards human conceptual understanding. In Section 7 we will discuss an idea we have to support an evolving taxonomy space that can represent the global conceptual relationships between resources.

This discussion is not intended as an indictment of FTP. Rather, our point is that FTP was designed at a time when point-to-point communication was the goal. The concept of global distributed collaboration involving widely distributed sites was simply not feasible when network bandwidth was so scarce. We also do not mean to imply that resource discovery cannot be supported with FTP. Rather, it is our intention to further develop this work, to support basic resource discovery among existing anonymous FTP archives, and also to develop a more sophisticated mechanism that might be used by archive sites in the future.

7. Conclusions and Future Work

Given increases in Internet bandwidth and inter-organizational computing, the time is ripe for significant advances in the technology that supports Internet resource sharing. Resource discovery is an integral part of the picture, since it is currently a limiting technology. We assume that far more resources exist than can be completely organized; that it will be more important to organize and locate some resources than others; and that it is difficult to reach global agreement over information sources and formats. Our approach is to allow incremental organization of the resource space, based on a range of different information sources of varying degrees of quality, ranging from high-quality descriptions of particularly popular resources to heuristically gathered hints about where a search for less popular resources might begin.

We have built a set of mechanisms that form an initial basis for resource discovery in the domain of public Internet archives accessible via the "anonymous" File Transfer Protocol. This is an interesting test case, because it encompasses a very large scale, administratively decentralized collection of resources, with considerable practical value. The prototype currently supports a range of activities corresponding to various levels of effort and levels of information quality that can be extended by a distributed collection of people contributing to the global organization of the resource space. The highest quality of information is derived from database support for cataloging the conceptual nature of individual resource repositories. Below that, caches are maintained to ease the task of finding resources that have been located in the past. Below that, the system monitors announcements of public archive availability from USENET electronic bulletin board articles, to provide a simple keyword-based attempted index of potential anonymous FTP sites around the global network.

An important aspect of our approach is that it makes use of existing infrastructure. We believe this technique will become increasingly important as widely distributed, heterogeneous networks proliferate. Even so, the technique has limitations compared to what could be achieved given an optimal set of protocols and services to use for gleaning resource information. We are therefore pursuing a hybrid approach, involving existing protocols to achieve rapid deployment among current archives, and a set of more sophisticated mechanisms for supporting resource discovery among future sites. We now describe these ideas.

At this point, we have plans for a number of areas of future work on the prototype. The first step is to improve some inefficiencies in the prototype (such as the size of the gdbm databases). Next, we will integrate the individual pieces of the prototype, to provide a unified tool interface. After these modifications, we hope to gain a large enough user base to try charting a significant number of Internet archive sites, to uncover problems and principles with this style of wide-area distributed collaboration. Deployment of this prototype will also allow us to experiment with a dynamic resource organization mechanism outlined in a recent paper, whereby related resources are clustered according to access patterns, to provide a dynamically evolving set of links between related resources [Schwartz & Wood 1990].

Next, we will extend the range of information quality levels to support finer-grained distinctions between information sources. Examples of the sorts of distinctions that can be made include the difference between moderated news groups, expert (peer) reviewers, archive curators, and voting.

The next step involves making the per-user-site caches available by anonymous FTP, so that users may share the map generated by the sum of each others' explorations. Such a map would likely grow to quickly cover a sizable proportion of the more popular Internet public archive resources. Of course, such a map raises privacy issues. There must be a way for users to specify that they do not wish their explorations to be recorded in the shared map. Moreover, the usual problems of cache consistency must be addressed. For this purpose we intend to use fairly simple techniques, such as timeouts and the detection of stale data upon use.

After these changes, we will explore a more sophisticated mechanism for describing resources, based on a Resource Description Language that can represent both conceptual descriptions and underlying concrete representations of data. While the conceptual descriptions will be used for resource discovery, the concrete representations can be used to support automatic translations among common data representation formats, and to allow independently catalogued data to be compared, using techniques from redundancy/reliability theory to improve the quality of the descriptions.

We are also interested in exploring a means of describing resources based on separate taxonomy and resource spaces. The idea is to allow the taxonomy space to evolve through a distributed collaboration process, and allow entries in the resource space to be *attached* to appropriate taxonomy space entries. This method will improve resource description uniformity, yet allow the organization to evolve smoothly.

As a longer-term goal, we plan to define a new Internet protocol that provides better support for distributed collaboration than can be provided by anonymous FTP. The chief advantage of anonymous FTP in our opinion is that it currently exists at a large number of sites, and is implemented on a number of different types of systems. Hence, it offers the possibility for rapid deployment of an interesting experimental system. However, as outlined in Section 6, anonymous FTP has a number of limitations for supporting global sharing. We plan to address these limitations using a protocol that is backwards-compatible with anonymous FTP.

Acknowledgements

We would like to express our appreciation to David Goldstein, Evi Nemeth, Panos Tsirigotis, and David Wood, who provided feedback on a draft of this paper. We would also like to thank the administrators of the many anonymous FTP sites around the world, without whose cooperative spirit this interesting resource

discovery test bed would not be possible.

8. References

[Arms 1989] W. Y. Arms. Electronic Publishing and the Academic Library. Paper presented to the Society for Academic Publishing, 11th Annual Meeting, May 1989.

[CompuServe 1986]
CompuServe. *CompuServe Information Service.* CompuServe Inc., Oct. 1986. Sales literature.

[Donahue & Widom 1986]
J. Donahue and J. Widom. Whiteboards: A Graphical Database Tool. *ACM Trans. Office Information Syst.*, 4(1), pp. 24-41, Jan. 1986.

[Fischer & Stevens 1990]
G. Fischer and C. Stevens. Information Access in Complex, Poorly Structured Information Spaces. Tech. Rep. CU-CS-461-90, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, Feb. 1990.

[Granrose 1990]
J. Granrose. Personal Communication. Cowell College, University of California at Santa Cruz, Sep. 1990. Electronic bulletin board posting containing Anonymous FTP site list.

[Horvath 1990]
S. M. Horvath. NSFNET Usage by Service. Message sent to nsfnet-reports@merit.edu electronic mail distribution list, Aug. 1990.

[Kahn & Cerf 1988]
R. E. Kahn and V. G. Cerf. *The Digital Library Project - Volume 1: The World of Knowbots.* Corp. for National Research Initiatives, Mar. 1988.

[Postel & Reynolds 1985]
J. Postel and J. Reynolds. File Transfer Protocol (FTP). Req. For Com. 959, USC Information Sci. Institute, Oct. 1985.

[Quarterman & Hoskins 1986]
J. S. Quarterman and J. C. Hoskins. Notable Computer Networks. *Commun. ACM*, 23(10), pp. 932-971, Oct. 1986.

[Robinson 1990]
G. Robinson. Personal Communication. Clarinet Communications Corp., Aug. 1990. Description of Clarinet communication service.

[Schatz & Caplinger 1989]
B. R. Schatz and M. Caplinger. Searching in a Hyperlibrary. *Proc. 5th IEEE Int. Conf. Data Eng.*, pp. 188-197, Feb. 1989.

[Schwartz 1989]
M. F. Schwartz. The Networked Resource Discovery Project. *Proc. IFIP XI World Congress*, pp. 827-832, San Francisco, CA, Aug. 1989.

[Schwartz 1990]
M. F. Schwartz. A Scalable, Non-Hierarchical Resource Discovery Mechanism Based on Probabilistic Protocols. Tech. Rep. CU-CS-474-90, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, June 1990. Submitted for publication.

[Schwartz & Wood 1990]
M. F. Schwartz and D. C. M. Wood. A Measurement Study of Organizational Properties in the Global Electronic Mail Community. Tech. Rep. CU-CS-482-90, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, Aug. 1990. Submitted for publication.

[Schwartz & Tsirigotis 1990]
M. F. Schwartz and P. G. Tsirigotis. Experience with a Semantically Cognizant Internet White Pages Directory Tool. To appear, *J. Internetworking Research and Experience*, 1(2), Dec. 1990.

[Vielmetti 1990]
E. Vielmetti. Personal Communication. Univ. of MI Statistics Dept., Mar. 1990. Discussion of method of curating comp.archives USENET news group.

[Wall 1989] L. Wall. *Manual Page for PERL - Practical Extraction and Report Language.* Jet Propulsion Laboratory, NASA, Oct. 1989.