Semesterarbeit in Wirtschaftsinformatik an der Wirtschaftswissenschaftlichen Abteilung der Universität Zürich

Organizing File Archives in Large Scale Distributed Systems

Martin A. Blatter
blatter@ifi.unizh.ch -or- blatter@icu.unizh.ch

Pfaffächerstr. 59
8913 Ottenbach
Phone: +41 (1) 761 20 02

Abstract

On the background of the constantly growing size and complexity of international networks the organization of public or file archives and archive arrays is getting more and more important. Based on statistical data of a selected *Internet anonymous FTP* archive, this study deals with some current problems of file archives in wide area networks. It presents mechanisms and solutions to reduce network traffic, costs, and overhead by improving transparency, consistency, and ease of use. Other topics being discussed in this paper are file replication, synchronization of distributed archives. Furthermore, it describes and comments on the protocols and tools currently in use and proposes requirements for possible new yet to be implemented protocols.

1. Introduction

The need for file archives arises when individuals want to share resources, e.g. source code, binaries, text, graphics, or music over a certain distance. There may be several users in a research institute or a large company working on the same project who need access to the same files stored on a dedicated local *file server*. In a larger scale, researchers at universities all over the world may be looking for scientific data from other researchers on a facility such as an *anonymous FTP archive* on the *Internet* or they might want to share their work with other people working on similar topics.

As competition in the 1990s will be rougher and tougher than before, there is a need for speed-up thinking. Products need to be developed, manufactured an delivered faster. This is what Kotler [20] calls a *Turbo-marketing strategy*. In multinational companies, or international enterprises with many affiliates, file archives can be used to give executives instant access to company internal data, such as marketing reports and sales figures. Other applications include the sharing of engineering data such as technical reports, drawings, or source code between several engineering departments. A well organized archive allows for long time storage of old information as well as easy updating of outdated data.

As long as the files are kept in a closed environment and used by a small number of individuals, there is not much need for a special organization of an archive. However, if the files are located on a publicly accessible server on a large international computer network, much more work has to be put in keeping the archive as consistent and user friendly as possible. Also, with the ever increasing size of files such as multi media data (graphics, animation, video sequences, music) it is becoming more and more important to prevent expensive long-distance transfers. This can be achieved by distributing the contents of an archive to several hosts at strategically important positions within the network topology. Another advantage of a distributed organization is the increase in availability and reliability of an archive due to the redundancy provided by systems sharing the same data.

This study will focus on the *Internet*, the world's largest computer network with a heavy academic background. However, the main findings apply to any wide area network of similar or smaller extent.

The Internet currently has a host population of more than 940,000 hosts [10, June 1992]. More than 242,000 hosts belong to the commercial (**`.com`**) top-level domain and more than 15,000 are located in Switzerland (**`.ch`**). Fig.1 illustrates the growth of the host population during the last eleven years.

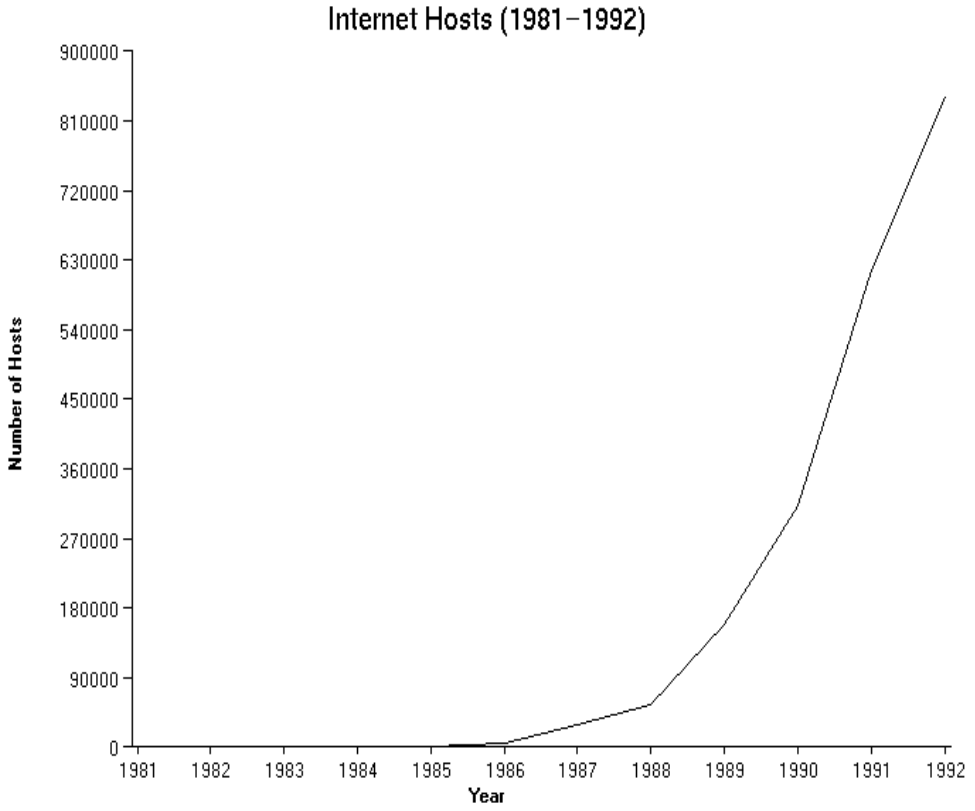


Fig.1 Growth of the Internet host population 1981-1992 (Source: [1] and [10])

As the Internet is based on the TCP/IP protocol popularized with the Unix operating system, most public file archives are based on the FTP software [7] originally developed at University of California, Berkeley. The goal of this work is to propose techniques to improve consistency and user friendliness amongst a single site or a whole set of sites sharing files with a mechanism of data replication without making the current set of tools obsolete.

File transfers are undoubtedly responsible for a large part of the traffic on the Internet [9]. A measurement study on the National Science Foundation Network (NSFNET) backbone by Claffy et al. [14] and more recent data found in [15] and [19] shows that in the last three years, ftp transfers accounted for 40% to 50% of all transferred bytes (Fig 2).

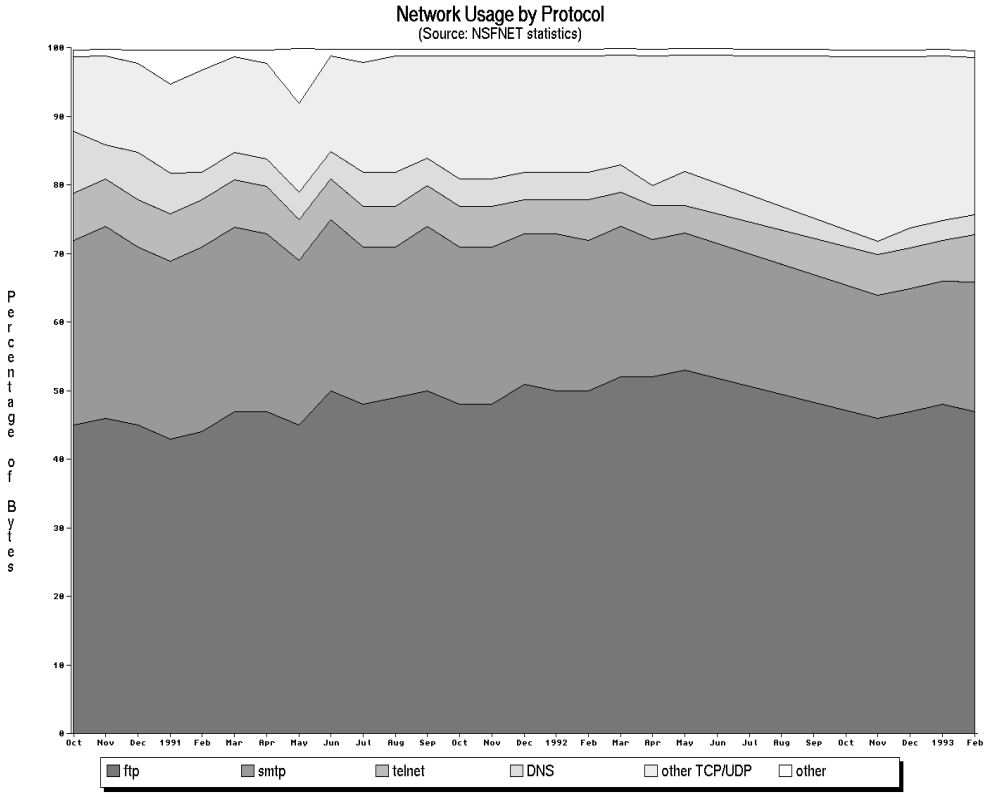


Fig. 2 Network usage percentage by protocol on NSFNET [14] from 1990 to 1993 based on data found in [15]

It is important to note that there is a discrepancy between the traffic measured in bytes and the number of packets transferred. Fig. 3 shows that interactive services like *telnet* or *irc* account for more packets than bytes, compared to the respective numbers for FTP and thus use network resources in a less efficient way.

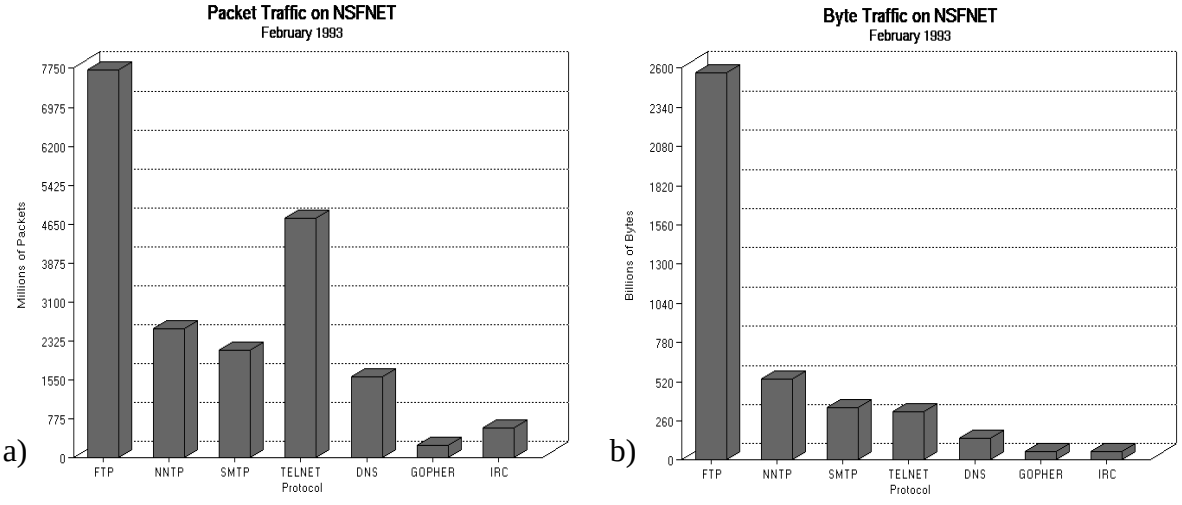


Fig. 3 Traffic on NSFNET measured in number of packets (a) vs. traffic in bytes (b)

The average size of an FTP *data* packets was found to be 350 bytes [14] (a full size packet contains around 576 bytes) whereas a telnet packet contains an average of about 70 bytes [14]. It must, however, be kept in mind that the size of an «interactive» FTP packet which does not carry data but rather protocol requests is usually about the same size as a telnet packet.

Fig. 4 displays the **packet** traffic on NSFNET in absolute numbers from 1989 to 1993. While these numbers are partly affected by changes in the NSFNET network structure and topology and measuring errors (1991) a long term trend can be well recognized.

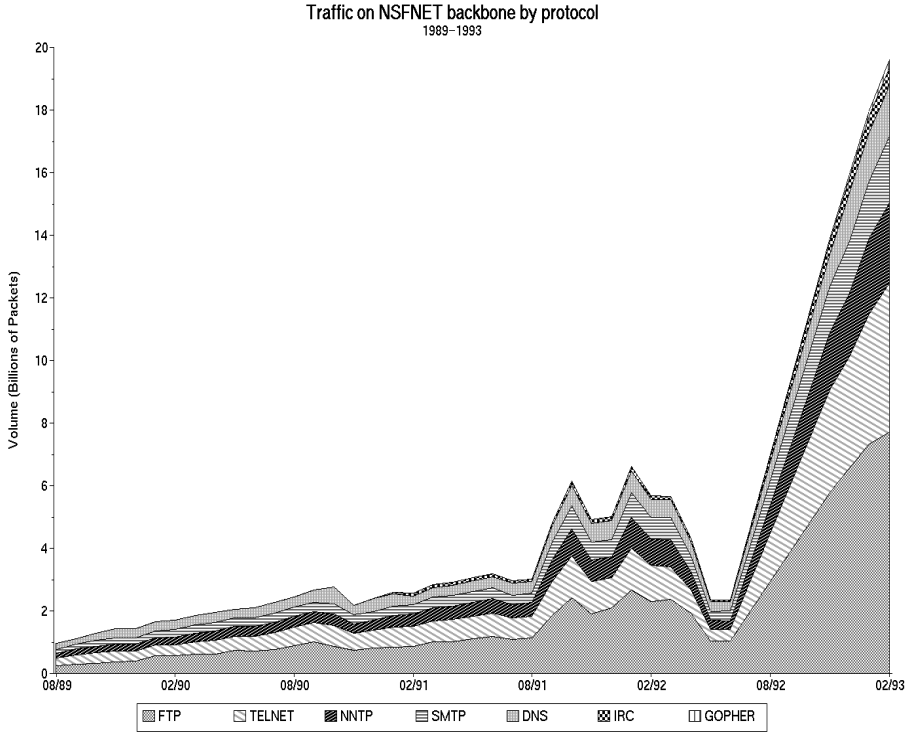


Fig. 4 Packet traffic on the NSFNET backbone by protocol (Source: [15])

The structure of the rest of this paper is as follows: In chapter 2 we will present an overview of some popular methods used today to retrieve information about desired files on the Internet. This process of «shopping around» for files which suit the needs of a user is called *Resource Discovery*. In chapter 3 we examine the requirements for an efficient file archive. Based on data of existing archives we will discuss several methods of organizing file archives and networks of archives. Chapter 4 and 5 describe an implementation of a large scale distributed Internet file archive initiated at the University of Zürich and our experiences after one year of operation. Finally, chapter 6 summarizes the main findings and concludes the paper.

2. Resource Discovery

Before connecting to a file archive, a user looking for a specific file has several possibilities to get the information he needs for retrieving it depending on what he knows about the file in question. The required information includes

- The full name or Internet address of the site where the file is located.
- The fully qualified path and file name on that host

There are of course several types of archive users:

Someone who is looking for a scientific paper or a specific source code usually knows exactly the name of the desired file or even where to look for it.

However, many users don't know the file name but they are aware of what the contents of the file is. They might be looking for documents, technical reports, or statistical data about a specific topic, or source code or binaries to solve a problem or accomplish a certain task. A short description of the contents would help them to judge if the file in question suits their needs or not.

There is also a large number of archive on the Internet users just looking for the latest freely distributed software or graphics without specific needs. It is probably the latter kind of user who wastes most of the network bandwidth. The resource discovery facilities presented in this chapter can only assist users with specific needs in getting the information and locating their resources.

2.1. Usenet

A commonly used method to announce the upload of a file to an anonymous FTP file archive is a posting in a Usenet [11] newsgroup. Table 1 presents some of the more popular newsgroups where such announcements are usually being posted. A study by Schwartz [2] found that in 1990, 89 different newsgroups regularly contain announcements. Therefore this can only be a small selection. A quick search through the news spool directories on the news server of the Computer Science Department at the University of Zurich revealed that the amount of announcements, requests and references to files available for anonymous ftp has grown to 4840 in 650 different newsgroups during two years. Of course, these numbers are not representative and they depend heavily on the amount of time articles stay on the server until they expire.

comp.archives	(Moderated newsgroup consisting only of upload announcements)
comp.sys.amiga.announce	(Files specific to the Amiga personal computer)
comp.sys.mac.announce	(Files specific to the Macintosh personal computer)
comp.os.ms-windows.announce	(Announcements relating to Windows)
comp.sys.next.announce	(Announcements related to the NeXT)
comp.windows.x.announce	(X Consortium announcements)

Tab. 1 Selection of newsgroups carrying upload announcements

Unfortunately, there are no binding rules on how such an announcement has to look like. It therefore greatly depends on the moderator of the respective newsgroup to introduce such rules and instructions and on the user to follow given rules. This makes it difficult to implement a program that is able to scan news articles for such upload announcements and extract the required information in a way like *aftpgather* [2] does. Another problem is that information gathered from news articles is subject to heavy inaccuracy and fast ageing. The reason for this is that many archive sites maintain a special *incoming* directory, where new files are being uploaded. To prevent misplaced uploads, the actual archive directories are usually read-only to the anonymous user. It is then up to the site administrator to move incoming files to the appropriate archive directory. The directory information of most announcements will therefore be obsolete as soon as the file has been moved to another place by the archive staff.

2.2. Archie

Another approach to reduce the efforts of searching for files is the Archie server [8] developed at McGill University of Canada. Archie is a comprehensive database of references to files on more than thousand (October 1992: 1020) anonymous FTP archives all over the world. The database can be queried interactively by connecting to an Archie server with telnet or non-interactively using the Prospero protocol [12].

Archie is very useful and efficient in finding a file as long as the user knows the name of it. Therefore, it is only of use to the first category of users as defined at the beginning of in this chapter. It is almost impossible to find out about the contents of a file by looking at its name. Many file names are short and cryptic due to the file name length limitation of some older operating systems and file systems and, as a consequence, often ambiguous. Also, many files on archive sites are packed and their file names consist of a compressor- respectively archiver-specific file name extension (e.g. `abc.tar.Z`). Short file names are also a common source of confusion to Archie users because it can happen that the same file name is used by completely different programs. By default, Archie outputs the names of all sites keeping the file regardless of their location within the network structure. So, users may be tempted to connect to a far away host with an expensive connection rather than retrieving the file from a nearby site.

Due to the large amount of sites maintained by Archie the database is only updated twice a week which is not enough to keep the database consistent with the state of the scanned archives.

Despite its drawbacks, Archie has become an indispensable tool for the resource discovery on the Internet and has been replicated recently in order to reduce the number of calls to the original Archie server.

2.3 Gopher

Many anonymous ftp sites also have *Gopher* servers installed. Gopher is a distributed document delivery system [13]. It allows users to quickly and easily browse through archive directories using a Gopher client program and to download files directly by selecting them from a list.

Thanks to its user interface, it is very easy to display **`README`**, **`INDEX`** and **`CONTENTS`** files with a gopher client. While this feature may help preventing superfluous downloads, Gopher generally makes transferring files almost too easy. A user may not realize how much network bandwidth such a transfer can waste. Fig. 5a-5c demonstrate a sample file retrieval session with Gopher.

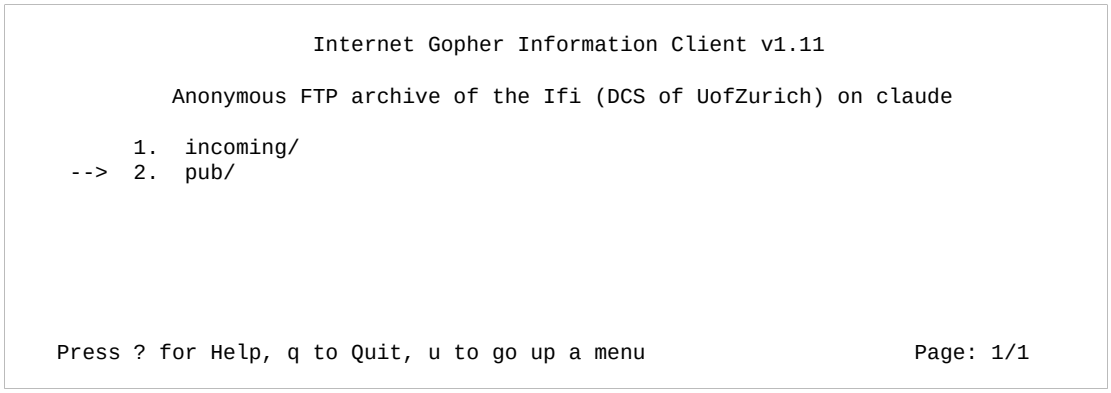


Fig. 5a Sample Gopher session: Selecting an archive directory (e.g. **`pub`**)

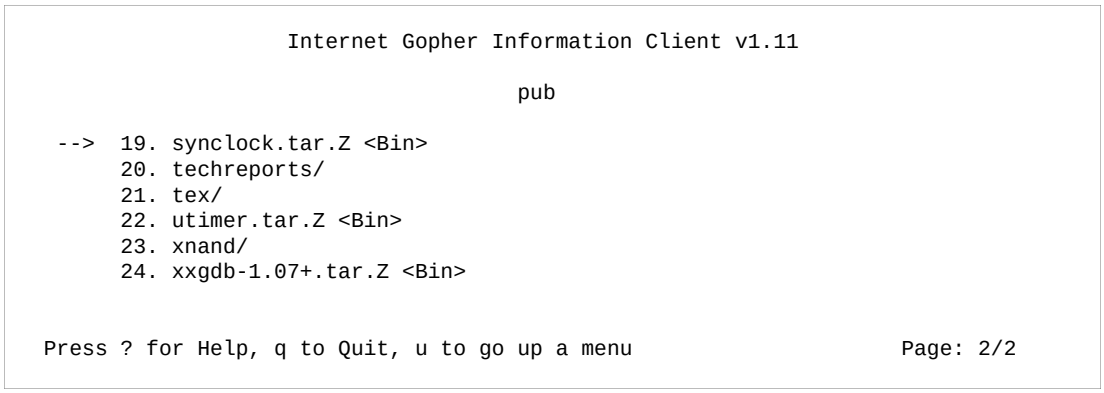


Fig. 5b Sample Gopher session: Listing files in a directory

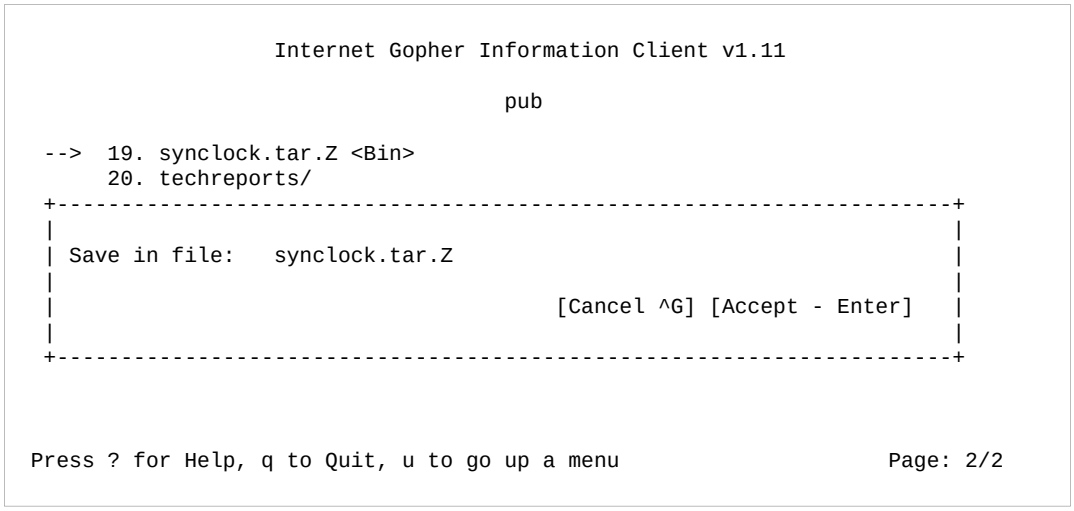


Fig 5c Sample Gopher Session: naming and retrieving a file.

3. Requirements

This chapter describes requirements, methods and mechanisms for an efficient network of file archives in large scale distributed systems. It presents and evaluates various methods for a better file archive organization.

3.1. Caching, Decentralization and Replication

As discussed earlier, the Internet file transfer protocol FTP is responsible for almost 50% of the Internet byte traffic. The main objective of decentralization, caching and data replication schemes is to reduce this load by diminishing the number of duplicate transfers of the same data and limiting the traffic to data exchange between adjacent hosts.

A **cache** in its purest sense is a write-through buffer (e.g. used on the memory bus of a computer). Here, we use the word cache to describe a facility (i.e. a disk on an archive) carrying replicas of files. It is up to a data **replication** scheme to create and maintain these copies or replicas on a cache. A cache can only contain a subset of all potentially replicable files. It has been observed that most Archie [8] requests are answered by using only a small part of the database information while some entries have never been requested. So, in this case it would be most efficient to replicate only the «popular» files using a cache. (See below for a discussion of the popularity of files).

Facilities carrying an exact replica of *all* the files of an archive are called **mirrors**. The term mirror is originated from a set of perl [17,18] scripts of the same name designed to replicate complete file archives on Unix systems [16].

According to Ewing et al. [3] almost half of all transfers could be saved by eliminating the number of duplicate transmissions, i.e. files that are being transferred more than once to the same destination host. An ideal archive organization would therefore have caches at strategically important locations on the network carrying important files. Unfortunately, as Maffeis [6] and Ewing et al. [3] discovered that a small subset of files is significantly more popular than others. While some files are being downloaded several hundred times, other files might never be downloaded. Distributing unpopular files to caches or other archives around the world would result in an unnecessary waste of resources. Maffeis found that 50% of all transfers related to only 12% of all retrieved files [6]. However, it is almost impossible to algorithmically decide if a file is of possible interest of users and subject to many downloads or not. No algorithm would be able to keep in account *all* factors which make a file popular.

One approach that has been brought up by some authors is to consider the **file type** in a caching mechanism. This may look problematic at first sight: Table 2 shows the contents of a directory of files with similar contents on an anonymous ftp server at the University of Zurich and the number of times every single file has been transferred. It shows that even files of the same or similar type can have totally different popularity.

file name	uploaded @	last d/l	byte size	downloads
bison-1.16.lha	Apr 15 1992	Dec 26 1992	406749	**53**
bmake15.lzh	Dec 31 1991	Dec 29 1992	135684	**56**
gasldsrc1.38.tar.Z	Dec 31 1991	Dec 25 1992	328897	**38**
gcc.2.2.2-diffs.lha	Oct 1 1992	Dec 24 1992	30823	**38**
gcc21-920420.lha	Dec 31 1991	Dec 24 1992	1691779	**31**
gcc222-fix1.lha	Aug 22 1992	Dec 29 1992	32720	**115**
gcc222-fix2.lha	Sep 11 1992	Dec 29 1992	7286	**132**
gcc222-fix3.lha	Nov 9 1992	Dec 30 1992	173155	**170**
gcc222.lha	Jul 13 1992	Dec 30 1992	3306066	**105**
gcc233.lha	Nov 19 1992	Jan 4 1993	3588223	**194**
gcc233libfix.lha	Dec 18 1992	Dec 18 1992	262229	**1**
libg++2.0-920319.lha	Sep 23 1992	Dec 29 1992	369705	**77**
gcctools-920420.lha	Apr 20 1992	Dec 29 1992	102158	**55**

Tab. 2 Files in the dev/gcc directory of icu.unizh.ch and their download counts

However, observing download counts for a large number of files over a longer period of time can give at least a clue to determine which categories of files are more important and which are negligible.

To find out if there really was a distinct correlation between file category and popularity, all outgoing file transfers on icu.unizh.ch were logged for three months. Instead of using file extensions (e.g. **`.jpeg, .Z, .dvi, .lha`** etc.) to distinguish between file types [3] we took a completely different approach: All files available on the server and uploaded during the three month period have been manually examined and assigned to one of the following 15 categories:

Biz	Business software (database, spreadsheet, word processing, DTP) including patches for commercial software
Comm	Communications and network software
Demo	Demonstration programs, presentations, slideshows
Dev	Software development tools, compilers, debuggers
Disk	Disk and file management utilities
Doc	Text documents, reports, newsletters
Ent	Entertainment software
Gfx	Graphics software and utilities
HW	Hardware projects
Misc	Miscellaneous files not falling in any other category
Mods	Sound modules, complete compositions and samples
Music	Music software
OS 2.0	Various utilities requiring new system software
Pix	Pictures, illustrations and graphs

During the observation period, 105,344 outgoing file transfers of 1,697 different files took place.

The «average number of downloads per file type» ratio used in Fig. 6 has been calculated as follows:

$$\text{average \# of downloads per file type} = \frac{\text{\# of times files of the file type have been retrieved}}{\text{\# of files of the file type}}$$

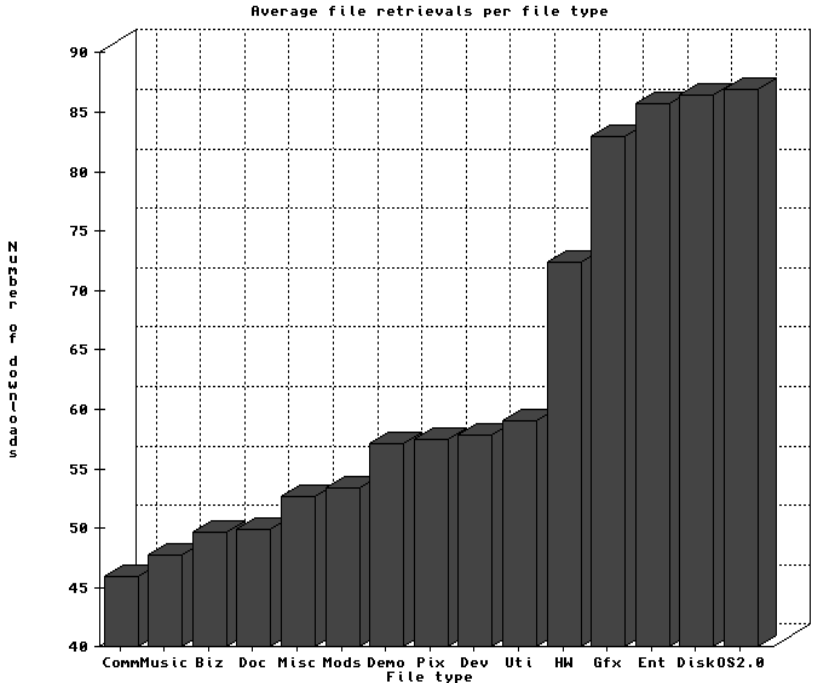


Fig 6: Average number of downloads on icu.unizh.ch by file type

Fig. 6 shows that the popularity of files on a file archive can indeed be tracked down to their file type: some categories have almost twice as many average retrievals than the least popular ones. A cache algorithm considering the file type to estimate which files would benefit from caching and which not would therefore use collected experience data to decide if a file should be made available to caches or not. However, keeping in account that even files of the least popular category have been retrieved an average of 46 times (with a standard deviation of 35), it would probably be more efficient in the case of icu.unizh.ch to put all files on cache hosts by creating mirrors carrying a replica of the **complete** file system of the archive. It is also

questionable if a simple replication heuristic such as «NAIVE» (as defined by Maffeis and Cap in [5]) which replicates objects only if they are accessed for the first time would be useful in this case considering the high average number of retrievals.

Another approach is to consider the file size. Maffeis discovered a distinct relationship between the size and the popularity of files: Smaller files have a significantly higher popularity than very large files [6].

3.2. Improving access time and resource discovery

Ewing et. al. found in their study [3] that only 44% of all FTP connections were resulting in an actual file transfer and suspected that most other operations were file manipulations such as listing directories. As discussed earlier, this interactive traffic uses considerable network bandwidth and, on the other hand, is painful for the FTP user. Although there are already many facilities designed to assist a user in the process of finding his resource such as those described in chapter 2, the Unix ftp utility is far from being user friendly and has many disadvantages. For example, it requires the user to switch from an ASCII transfer mode, which is turned on by default, to a special binary transfer mode. The ASCII mode is used to convert text files to an intermediate «standard network format» which can be used to compensate for different character sets on different operating system platforms. Binaries transferred in ASCII mode are corrupted and therefore unusable and must be retransferred. This misfeature alone is the cause for many duplicated transfers [3].

There is a wealth of simple methods to reduce and assist browsing (i.e. changing directories, listing directories) *on an archive*. First, the directory tree structure must be logically arranged and predictable. It should not contain too many subdirectories but at the same time the number of subdirectories in a directory should not be too large. Many archives have directories for every computer architecture they support in their archive top level. The next directory level usually contains subdirectories for the most important file types which themselves contain subdirectories which divide file types further. All files except some special **`README`** or contents listings are exclusively in the lowest directory level. Second, incoming files uploaded by users should be separated from the real archive to avoid confusion because of misplaced files. New files should be examined by an administrator and moved into the appropriate directory according to the file type.

Index files are also a very important feature of a well-managed archive. They should be available in a compressed format for users who want to copy them to their local machine for closer inspection. A readme file in the top-level directory should inform users about the archive contents directory structure, and other administrative details. It is also useful to display a brief notice about the contents whenever the user changes to a directory using the **`cd`** command to make sure the user does not list a directory he is not really interested in.

Many attempts have been made to enhance the existing ftp utility by making it easier to use, or by adding additional commands beyond the scope of the RFC specifications [7] to the ftp

server to make resource discovery easier. Other approaches, while still based on the ftp protocol, try to replace the user interface of the ftp utility.

3.3. Keeping Consistency

A crucial point for distributed file systems is consistency between the server and its caches. First, the cache or mirror system should contain the same files as the server, second, the files must be located in the same directories as on the replicated site. Inconsistent file system organization can be frustrating to users and may even have the effect that the effort for resource discovery (directory browsing) is much higher.

Also, the file systems must be synchronized regularly. The files must already be available on the cache system before the first upload announcement (e.g. Usenet) reaches the user. Users of a mirror site should not get conscious of the «propagation delay» required to distribute a certain file from the replicating site to the respective host. Otherwise they might be tempted to rather connect to the site where the file was first uploaded instead of using the local cache or mirror.

For example, when version 5 of the X Window System (X11R5) was released by the MIT consortium, it was made available on more than 20 anonymous ftp sites around the world by manual replication before it was publicly announced on the networks in order to prevent a network congestion.

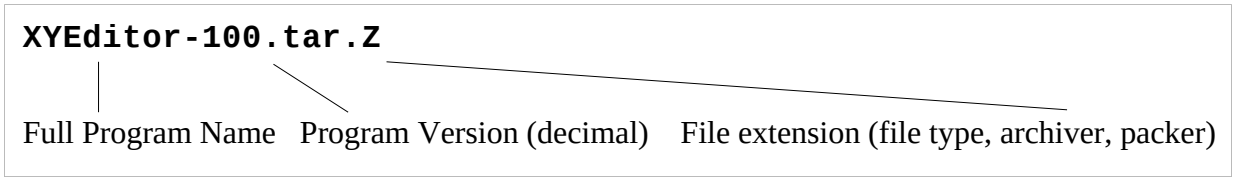
3.4. Managing Updates and Duplicates

Sources, binary files as well as text files are often updated by their authors to fix bugs or to add new information. They replace files that already exist on the archive. In most cases it makes no sense to keep older versions. However, sometimes, old versions are still required, especially if the new revision is supplied as a patch which must be applied to the original file. Because all files uploaded with anonymous ftp get the same file ownership information it is not possible to identify their originator. Therefore most archives don’t allow anonymous users to delete files. If someone provides a new version of an existing file the only thing he can possibly do is to leave it up to the archive’s administrators to delete the old copy. Another problem is that many archive users don’t name different versions or patches of their files in a consistent way. Also, users could independently upload the same file twice with different file names. This leads us to another important issue:

3.5 File Naming

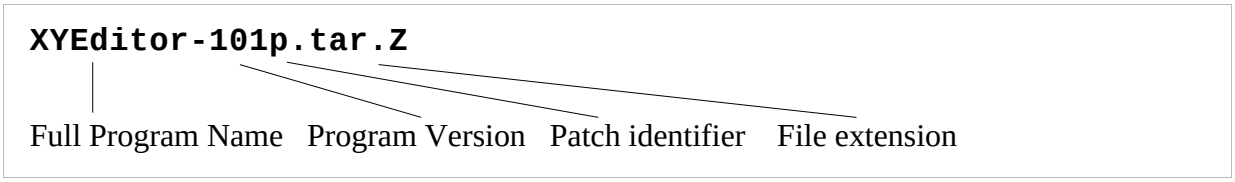
To make resource discovery (i.e. archive searches), update and duplicate management easier, archive administrators should enforce certain naming conventions. Unfortunately, some operating systems (e.g. MS-DOS or VMS) have restrictive file name length and structure limitations which require users to make their file names short and cryptic.

Here is an example of how a simple file naming system could look like:



The full program name allows to identify the contents of the file without having to take a closer look at the contents of the archive or a readme file. The decimal program version identifier (where 100 corresponds to version 1.0, 122 equals 1.22 etc.) allows for easy sorting of different versions of the same file

A patch update could use the same program name as the file it is based on but uses a different version number and a «p» character identifying it as a patch archive, e.g.



For **README** descriptions or announcements accompanying the document or package, the string **readme** would replace the file extension:

```
XYEditor-100.readme
```

Enforcing a naming scheme requires a lot of discipline by the archive users but our experiences presented in chapter 4 prove that it is indeed possible to «educate» archive users to take care of file naming conventions.

4. Case Study: Aminet

The following chapter describes an implementation of a replication scheme initiated at the University of Zurich. It was designed to solve the common problems of a traditional stand-alone anonymous ftp servers by decentralization and the use of simple but efficient file distribution. The project is called *Aminet* because it originally concentrated on binary files for one personal computer architecture, the *Amiga personal computer*. Today, Aminet carries other files as well (e.g. scientific documents, Unix source code etc.).

4.1. Philosophy

Aminet has been established in 1992 by members of the ICU, the Computer Science student's association at the University of Zurich which is running a bulletin board system and a file database for its members on two Unix workstations. The main motivation behind the Aminet project was first to provide a well managed *local* file base to prevent users of local Unix accounts from transferring files from remote sites themselves. It has been observed that before the introduction of the local file base, in many cases different users were downloading the same files independently. The local file archive should be also accessible by users of the ICU bulletin board system. Furthermore, we discovered that at that time there existed one large Internet ftp server in the United States carrying binaries and sources for the Amiga personal computer (ab20.larc.nasa.gov) and interested users from everywhere in the world were connecting almost exclusively to this site. When this archive suddenly had to close down, we decided to replace it.

Based on our observations and the experiences with ab20.larc.nasa.gov, the most important goal was to use the network resources as sparingly as possible. We realized that most of these problems could only be solved by a large network of sites carrying the same file base and exchanging files using a replication scheme. We decided to create a concept that keeps network load down and at the same time tries to address most of the other problems discussed earlier in this paper (Tab. 3).

Save resources by:
- creating a network of archives around the world
- replicating file systems
- supporting resource discovery
- simplify file management and administration

Tab. 3 The four main goals of the Aminet project

The system should be based on existing TCP/IP software and run on any *Unix* system to make it possible for all users to immediately take advantage of its features. Table 4 shows the five levels on which Aminet software is based.

- User Interface
- Statistical tools
- File management tools
- Administration tools
- Enhanced ftpd (FTP server)

Tab. 4 The five software levels of the Aminet system

The structure of the Aminet network could be described as a *hierarchical single-directional mirror system.* This means that there is one master host (called *Aminet Server*) where users put their files and an arbitrary number of *Aminet Mirrors* which mirror the archive file system of the Aminet Server by polling it regularly. The server-mirror structure has been chosen to prevent multiple uploads of the same file to different mirrors and to make administration easier because the administration work has to be done on the server only. Table 5 shows the 12 hosts currently participating in the Aminet project. The client-server structure of Aminet is displayed in Figure 7.

Location	Host name	IP Address
Switzerland	icu.unizh.ch	130.60.80.80
Switzerland	litamiga.epfl.ch	128.178.151.32
Scandinavia	ftp.luth.se	130.240.18.2
Germany	ftp.uni-kl.de	131.246.9.95
Germany	ftp.uni-erlangen.de	131.188.1.43
Germany	ftp.cs.tu-berlin.de	130.149.17.7
Germany	ftp.th-darmstadt.de	130.83.55.75
Germany	ftp.uni-paderborn.de	131.234.2.32
USA	wuarchive.wustl.edu	128.252.135.4
USA	merlin.etsu.edu	192.43.199.20
USA	oes.orst.edu	128.193.124.2
Australia	splat.aarnet.edu.au	192.107.107.6

Tab. 5 The 12 Aminet sites

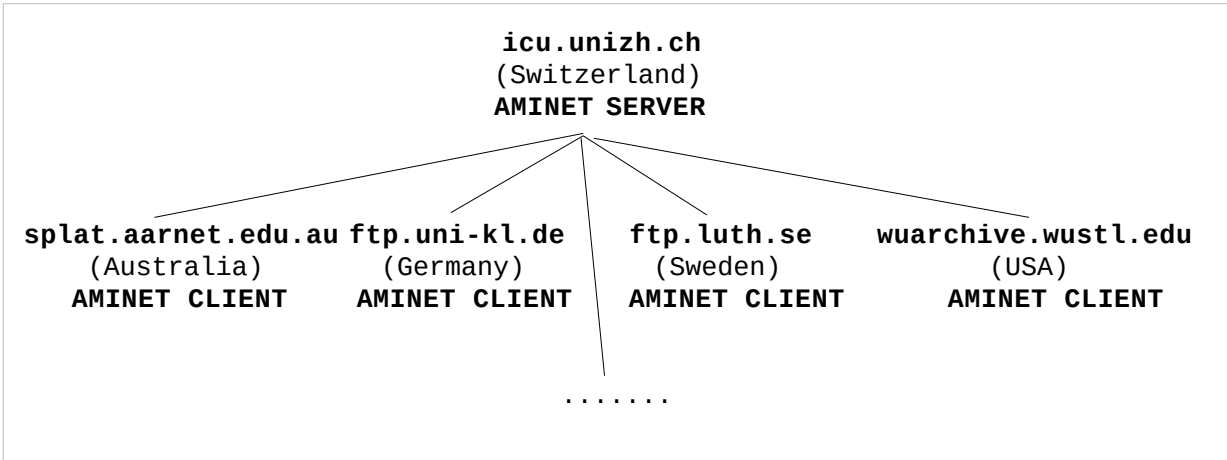


Fig. 7 The hierarchical client-server structure of the Aminet network (not comprehensive)

4.2. File Organization

In order to keep the archive consistent and well-managed, the file structure on the Aminet is divided in two main directory trees and follows the traditional structure of most current ftp sites as described in chapter 2.1. One tree consists of a directory named **new** which is writable by anonymous users and where new files can be uploaded. The other tree consists of two directory levels containing the read-only archive (see Fig. 8). Aminet currently only supports one machine architecture for binary files (i.e. the Amiga) but could easily be expanded for more architectures by introducing an **/pub/aminet** directory tree for every machine type.

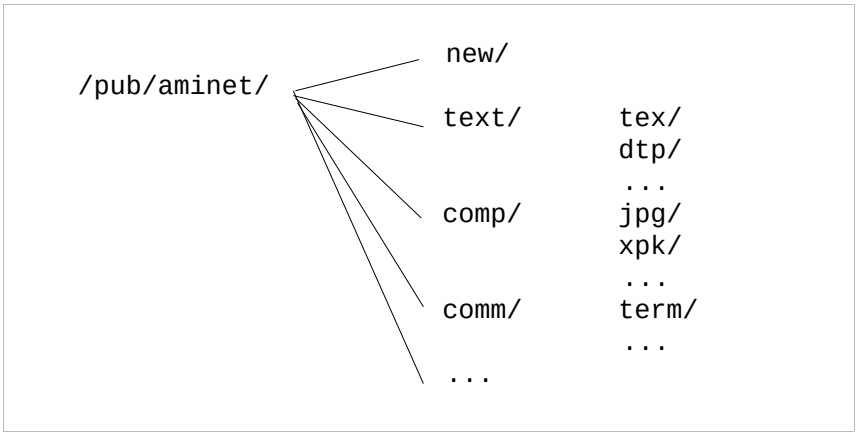


Fig. 8 The Aminet archive file structure

4.3. Special Files

Various additional files are required for the mirroring system to operate properly. Some files must be created by the contributor of the file and require quite an amount of discipline from the single user. A file named **README.BEFORE.UPLOAD** located in the **new** directory lists these rules some of which are discussed in detail below. It also proposes non-stifling naming conventions to ease administration, duplicates management and resource discovery. However, most files are maintained automatically by the administration tools.

4.3.1. Files created by the archive user

Every user uploading a file to the Aminet server is expected to provide a corresponding **README** file which describes the contents. It primarily serves the site administrator to judge where in the **/pub/aminet** tree he should put the file. There are no rules whatsoever on how the **README** announcement should look like. However there is one short line that should be in every such announcement: It starts with the keyword **SHORT:** followed by a 40 character summary of the file's purpose. The Aminet file management utilities scan every announcement for that particular keyword and use the information to create various files described in the next chapter. The name of this **README** file is supposed to consist of the name of the archive or file it refers to and an extension named **.README**, e.g. **FOO.README**

4.3.2. Files created by the Aminet file management and administration tools

The Aminet file management and administration tools are a collection of small C-Shell scripts handling all tasks required to maintain file lists and statistical data. They also take care of updating the mirror files required for the Aminet replication system.

4.3.2.1. File Database

The ftp daemon program on the Aminet server maintains a database of all files on the archive in GNU **gdbm** database format containing various statistics like the date of upload, the date of the last download, the number of times the file has been retrieved and the e-mail address of the person who contributed the file. All administration and statistical tools are built around this file database.

4.3.2.2. User files

User files are text documents designed to assist the user in the process of resource discovery.

4.3.2.2.1. RECENT

The **RECENT** file located in the root of the **/pub/aminet** tree contains a listing of all files that have been uploaded within the last 7 days sorted by age. The information of this listing consists of the file name, its location within the **/pub/aminet** directory tree and the short description given by the user in the **README** file. The **RECENT** listing is prepared automatically twice a day. Two versions of the same listing are being prepared, one in plain ASCII format another in a compressed format (using the Unix **compress** utility). Table 6 shows an example of how the **RECENT** listing may look like.

```
# Recent uploads to icu.unizh.ch [130.60.80.80] on 16-Oct-92
# These are the last 7 days' uploads, newest first.
#
#File                 Dir        Size Description
#-------------------- ---        ---- -----------
emath-3d.V1.0.lzh    gfx/misc    35K 3d function plotter
AmigaJPEGV3.lha      gfx/conv   138K Version 3 of the JPEG software
sas2ced5.lha         dev/c       10K interface from SAS/C 5.10b to CygnusEd 2.12
PCQ12src.lzh         dev/lang   149K PCQ Pascal version 1.2 (source)
Din.zoo              dev/misc    31K Interprocess communication library
shadow4.6.lzh        dev/misc   173K concurrent-object-oriented additions to AmigaOS
sana2.lzh            dev/misc    48K Official CBM network architecture standard
TRIX.lha             dev/misc    12K Complete Amiga Neural Net package.
AmigaPGP20.lha       util/crypt 444K Public-key encryption
ARexxAppList-10-92.l util/rexx   25K The ARexx Application List, 10/92
```

Tab. 6 The **RECENT** file

The **RECENT** listing has mainly been introduced to address the problem of archive users without specific needs discovered by Ewing et alt. [3] and discussed in chapter 2. It prevents them from scanning directories by just changing directories, listing and downloading files of potential interest. Thanks to the short description, it also makes it easier for the user to decide if the file in question is of any use for him. So it comes to no surprise that the **RECENT** file has become the most downloaded file on all Aminet sites. On the Aminet server, there are currently almost as many retrievals of the **RECENT** file as there are successful logins.

4.3.2.2.2. SHORT, ls -lR.Z, CHARTS

The **SHORT** file, also located in the root of the **/pub/aminet** directory tree, is somewhat similar to the **RECENT** listing. However, it contains entries for *all* files in the archive sorted by file name instead of date. Like the **RECENT** file, it includes short summaries of the file contents. The listing is updated once a day. As it can grow quite large it is only available in a compressed format.

ls -lR.Z is named after the command sequence used to generate it. It is a compressed version of a recursive listing of all files on the archive site. The file can be downloaded by any user but its main application is to provide the Archie database, which polls the Aminet server and most clients on a regular basis, with detailed information about the contents of the site.

Another listing which is mainly of informal use for archive users is named **CHARTS**. Similar to the **SHORT** and **RECENT** files, it contains a location indicator and short descriptions. However, unlike the other files, it is sorted by the number of times the entries have been downloaded.

4.3.2.3. Mirror files

These are the files that make up the core of the Aminet replication scheme. Each client system has its own mirror file in a special **/priv** directory which is not readable by normal archive users. It consists of a list of simple *get* commands for the Unix ftp client software, one line per file to be transferred. The mirror files are maintained by the administration scripts. Only files that have already been moved to their final location in the archive directory tree are listed. Mirror files can also contain delete and move instructions for the client.

If the connection between the server and the client was successful and all file transfers have been completed without error, the entries for the transferred data will be removed from the mirror file.

This system has several important advantages over the mirror package [16]. Most noteworthy, the client does not have to list the complete contents of the server to compare and synchronize file systems every time it connects to the server because the server itself keeps track of which files were replicated and which not.

4.4. The Administrator

Administering an Aminet client site is quite easy, since most of the painful daily work to maintain the archive is done on the server machine. Of course, client sites must strictly adhere to the directory tree specifications of Aminet and have the replication and polling scripts installed. Much more work must be done on the Aminet server: The administrator has to examine every **`README`** document in the **`new`** directory and move it and its associated files to their final destination within the archive tree. The scripts creating mirror files for clients can be both run by a crontab entry or manually by the administrator after changes to the file system have been done.

4.5. Replication Techniques

The file systems on server and client machines are currently synchronized on a regular basis, depending on the client's capacity and needs by polling the server machine. A special feature in the Aminet ftpd (ftp server) program used on the server allows clients to log in with special privileges and therefore circumvent the user count restriction. As the synchronization process is fully automated it can be activated from within a crontab at times when network activity is low and transfer rates are reasonably high. Of course, these times greatly depend on the location of the respective client within the Internet network topology and it is up to every client administrator to find the best configuration for his situation.

It has been observed that synchronizing file systems about twice or three times a day is sufficient for most clients keeping in account the propagation delay of announcements submitted to Usenet until they reach the users.

4.6 File Expiration

File space on archive sites is often limited. Adding new mass media devices to a system does not solve the real problem, namely that many older file are simply not worth to be stored on the archive because no user will ever download them again. Also, a partition where files can be uploaded should never be full in order to prevent incomplete transfers and not frustrating users. To overcome this problem, a *file expiration mechanism* was introduced at icu.unizh.ch.

The file database on icu.unizh.ch provides useful information for an algorithm to decide if a file is still worth staying on the archive. The expiration algorithm takes two parameters: First, the number of retrievals (used to determine if a file is popular or not). Second the age of the file. The higher priority was given to the age of the file. Every file has a chance to stay on the archive for at least a certain amount of time until its popularity is even been considered. Depending on the amount of free space required on the archive file system, the minimum number of downloads required to justify the existence of a file can be raised or lowered.

4.7. User Interfaces

4.7.1. ftp client

The ftp client program offers a very simple way of interacting between the user and an Aminet archive host. Although the ftp server (ftpd) used on icu.unizh.ch includes several so-called site-extensions not found in the RFC specifications on which all ftpd implementations are based on, it is still not easy to use. Some of the more important extensions are presented here:

```
quote site xcat
```

Dumps the contents of ASCII documents regardless of if its archived and compressed or not. Many compression schemes are supported including Berkeley Unix Compress (**.Z**), Lempel-Ziv, Huffmann (**.lzh, .lha**), Zoo etc. (Tab. 7)

```
ftp> quot site xcat make-3.63.readme
 213- Contents of ‘make-3.63.readme’ follow:
   Short: Amiga port of GNU make, OS2. only
   Uploader: bg@macrohard.com
 213- Contents of  ‘make-3.63.readme’ complete.
```

Tab. 7 Output of the quote site xcat command

```
quote stat
```

Lists the contents of compressed archives along with several statistics about who uploaded the file and how much it has been downloaded (Tab. 8). The actual listing of the files in the archive depends on the archiver used for the particular file.

```
ftp> quot stat make-3.63.lha
211- status of make-3.63.lha:
211- Created Thu Jan 28 22:03:52 1993 by bg@macrohard.com.
211- Last downloaded Tue Mar 30 04:20:48 1993, total download count 110.
  PERMSSN  UID GID  PACKED    SIZE  RATIO  CRC       STAMP       NAME
 ---------------- ------- ------- ------ ---- ----------------- -------------
 [generic]           60396  112488  53.6% e957 Jan  9 13:40 2009 make-3.63/bin/make
 [generic]            1203    2308  52.1% 41e9 Jan 27 17:36 1993 make-3.63/build.sh
 [generic]            1212    3512  34.5% bcce Jan 28 18:33 1993 make-3.63/config.h
 [generic]            6993   17982  38.8% dc3e Jan 28 18:39 1993 make-3.63/COPYING
 [generic]            9285   25265  36.7% 56b8 Jan  3 05:02 1992 make-3.63/libs/COPYING.L IB
 [generic]           77035  148428  51.9% 10da Jan 28 19:28 1993 make-3.63/libs/ixemul.library
 [generic]             693    1781  38.9% 8d41 Jan 28 18:47 1993 make-3.63/make.diff
 [generic]            3175    7992  39.7% 3acd Jan 28 19:23 1993 make-3.63/Makefile
 [generic]            1618    3167  51.0% 90dd Jan 28 22:01 1993 make-3.63/README
 ---------------- ------- ------- ------ ---- ----------------- -------------
  Total    9 files  161610  322923  50.0%      Jan 28 22:26 1993
211 End of Status
```

Tab. 8 Output of the quote stat command

Furthermore, the ftpd automatically detects if a user tries to transfer a file containing binary data in ASCII transfer and issues a warning message. Also, it supports the «resume transfer» feature for incompletely transferred files.

4.7.2. adt

ADT (Aminet Download Tool) is a user-friendly front end to the ftp program which uses the special files and features of Aminet client hosts. ADT is implemented with the C language. Its user interface, created with the Unix curses package, is loosely based on the popular freely distributable electronic mailer program «elm». It allows the user to directly select the files from lists containing file names and descriptions. These lists can be sorted by name, date, size and age. ADT includes a pager to display README files and other documentation and list the contents of archived or compressed files. Desired files can simply be selected (tagged) by moving a highlighting bar with the cursor keys. A special «batch download» feature helps downloading several files or documents at once (Fig. 9a to 9d). ADT also includes an interactive search feature

The advantage of a tool like ADT is that users do not have to worry about the directory structure and complex ftp client commands. It also takes care of the ASCII mode problems of the ftp protocol. However, the same that has been said about gopher also applies to ADT: While it greatly assists the process of resource discovery, it makes downloading files very easy - almost too easy.

```
New files: 132/142         Aminet Download Tool 1.2.4                Page: 9/10

 File (by dir)        Dir         Size Description                        Readme
 NTSC4NTSC_V2.2.lha   os30/util    16K Opens all Screens in NTSC (AGA-Support)
 intelclip.lha        pix/misc     11K "intel outside" logo for PDraw 3.0      *
 xmen.lzh             pix/misc    505K XMEN cartoon                            *
 conefog.lha          pix/trace   184K Raytraced Pic created in Imagine2       *
 5min0320.lha         text/anno    13K GEnie 5-Minute News - 3/20/93
 ar101.lha            text/anno    54K Report - #1.01 3/19/93
 cebit_review.lha     text/anno   121K CeBit review                            *
 nenscript1.3.lha     text/misc    56K PD enscript clone  (text->PostScript uti*
 MultiPrint15.lha     text/print   29K MultiPrint15 - Nice output of text files*
 ppmore20.lzh         text/show    55K File viewer, OS2.x or greater           *
 unz51d3xi.lzh        util/arc     80K UnZip version 5.1d3 from Info-Zip for un*
 gzip1.07.lha         util/gnu     45K Unsupported port of GNU zip, gzip1.0.7
 daterecall16.lha     util/misc    11K Scheduler for birthdays, exams, dentist,*
 dbb11.lha            util/misc   116K GUI Digital Logic Circuit Simulator (WB2*
 PPI_040.lha          util/misc   264K 040 software that comes with Zeus board *
 PCHGLib12.lha        gfx/misc    598K PCHG (palette change) chunk specificatio*

             d)ownload i)nfo l)ong n)ame r)eadme s)ort v)iew q)uit

Command:
```

Fig. 9a Listing **RECENT** files in short format with ADT

```
  Name PCHGLib12.lha        Date Tue Mar 23 18:21:10 1993     Size 612406

Short: PCHG (palette change) chunk specifications, tools and examples.

This archive contains the PCHG (Palette CHanGe) IFF chunk specs and
tools. The new IFF chunk PCHG allow to specify line-by-line palette
changes in a simple way which is independent of the video mode; while
it allows up to 65536 registers, it's usually shorter of an equivalent
CTBL or SHAM chunk. Library code with full source and documentation is
provided for a straightforward implementation in your programs. PCHG
has been developed in BIX through an open discussion of many Amiga
programmers, and it is our hope that it will become the Amiga standard
for palette change technology. Stunning sample pictures included 8^).









                    End of file - press space to return
```

Fig. 9b Displaying a **README** file

```
All files: 2069/2694       Aminet Download Tool 1.2.4              Page: 138/180

 File (by name)       Description                                          Readme
 Scale.lha            musical scales                                         *
 SCAN8800.lzh         Database for shortwave frequencies                     *
 schelober.dms        Database manager                                       *
 sci.3d-illusion.lha  Optical illusion
 sconfig.lha          Some DiskMaster config files                           *
 scrammer.lha         RAM control (2.0 required)
 SCRAMMER373B.lha     Check and set CPU & SCRAM options                      *
 SCRAM_2000.lha       8MEG/SCSI Controller KitWare
 SCRAM_500_KITWARE.LZ 8MEG/SCSI Controller KitWare                           *
 ScreamForHelp.lha    Musical score
 ScreenSelect_V1.2.lh Commodity to change screen orders.
 scripts.lzh          Unix shell scripts
 ScrnTst.lha          Program to test monitor quality, v2.0                  *
 SCSIMounter203.lha   Version 2.03 of SCSIMounter. OS 2.0+ only
 ScsiTape.lzh         Scsi-Direct tape handler                               *

             d)ownload i)nfo l)ong n)ame r)eadme s)ort v)iew q)uit

name-i-search: SCSIMoun
```

Fig. 9c Listing files in long format and searching for strings

```
Tagged: 2/2               Aminet Download Tool 1.2.4                Page: 1/1

 File (by nothing)    Description                                        Readme
+ScrnTst.lha          Program to test monitor quality, v2.0                *
+SCSIMounter203.lha   Version 2.03 of SCSIMounter. OS 2.0+ only            *









               d)ownload i)nfo l)ong n)ame r)eadme s)ort q)uit

Command: Download

                                  Processing....
```

Fig. 9d Display and retrieve tagged files

4.7.3. DCC

Another popular interface to the Aminet archives is IRC. IRC is the Internet Relay Chat service which allows Internet users to communicate with each other in different conferences. IRC has a built-in efficient file transfer protocol called DCC which can even be used during a conversation. The Aminet IRC client, also implemented in C, allows IRC users to transfer files using the DCC protocol. It also assists users in locating files, either globally (using an archie front end) or locally by querying the local file listing. It can output descriptions and information about the size of the queried files. As shown in Fig. 10, it allows to search not only for file names but also for expressions in the description line. Every Aminet client can have its own IRC front-end and some clients offer additional resource discovery features.

```
/msg Merbot help
-> *MerBot* help
-Merbot- This is the IRC frontend to the merlin.etsu.edu ftp site. Commands:
-Merbot- INFO     Sends by /dcc a complete help file
-Merbot- LIST n   Shows last n uploads, default 10, max 20
-Merbot- NEW      Sends by /dcc the list of last 14 days' uploads
-Merbot- FIND str Finds a file on merlin.
-Merbot- ARCH str Does a worldwide archie query for str. 3 replies max.
-Merbot- GET  str Sends a file to you. No path required.
-Merbot- Please note that merlin.etsu.edu also has a CDROM of the ab20
-Merbot- archives online, as they were when they closed.  FTP over and
-Merbot- check it out!
-> *MerBot* list 3
-Merbot- FILE              DIR        SIZE DESCRIPTION
-Merbot- conner-hd-stats.z hard/anno  225K Specs for Conner Peripherals HD.
-Merbot- xdrop2.21.lha     util/pack   23K Version 2.21 of xdrop. Requires xpk
-Merbot- AmigaScope.lha    hard/hack   25K 8 channel digital oscilloscope, use
-> *MerBot* find xpk
-Merbot- FILE              DIR        SIZE DESCRIPTION
-Merbot- xdrop2.21.lha     util/pack   23K Version 2.21 of xdrop. Requires xpk
-Merbot- xpk24usr.lha      util/pack  142K Compression package, user's edition
-Merbot- xpk24dev.lha      util/pack   94K Compression package, developer's ad
-> *Merbot* get xdrop2.21.lha
*** DCC SEND (xdrop2.21.lha) request received from Merbot
*** DCC GET connection with Merbot established
*** DCC GET:xdrop2.21.lha connection to Merbot completed
```

Fig. 10 A sample IRC/DCC session

4.7.4. Usenet Postings

To assist resource discovery the statistical tools automatically place weekly postings in the Usenet newsgroups *comp.sys.amiga.archives* and *de.comp.sys.amiga.archives*, which have been created for this purpose. The announcement contains a list of the most recent files uploaded to Aminet, similar to the one found in the **RECENT** file, along with a table of all Aminet sites and short communications of the archive staff (reminders, announcements etc.). Other non-Aminet sites (e.g. nic.funet.fi) have also started using these newsgroups.

5. Experiences and improvements

The experiences gathered during one year of operation of the Aminet project were very encouraging. While most technical problems could be solved immediately, some drawbacks appeared only after several months. In the beginning, our biggest concern was the lacking discipline of many users. Often, files were uploaded without accompanying **`README`** file or the documentation did not include the keywords required for our administration software. This could be resolved partly by including short upload instructions in the weekly «recent uploads» posting and a monthly FAQ (Frequently Asked Questions) document on Usenet explaining how to use anonymous ftp on the Aminet. Also, short instructions were added to the «message-of-the-day» text which appears on every anonymous login. Users who were still contributing files with insufficient documentation were sent a pre-fabricated instructions file by e-mail. As a result, after a few weeks, almost all contributed files were accompanied by **`README`** files. Still, some users did not care about providing documentation to their upload. Probably, only an enhancement of the ftp protocol for allowing the server to interactively ask for a description for every uploaded binary file (which would then be stored in a **`README`** file) could solve this problem.

Recently, a few additional optional keywords for the **`README`** file were proposed. They will allow future versions of the administration and file management utilities to automatically move new contributions to the appropriate archive directories and help the administrator to find out the appropriate file category.

Another concern was that, although users were mostly following naming conventions, updates were very hard to handle. Often, two versions of the same file were categorized differently by the archive administrators and duplicates went unnoticed. Also, it was sometimes impossible to distinguish between incremental updates, patches and replacements. These problems have only partly been solved yet and will still require much additional research.

It was noted after a few months that some users were always connecting to the server, regardless of the location of their host within the network topology, instead of using a local mirror. We discovered that the main reason for this behavior was the propagation delay and the lacking consistency between server and clients. As a consequence, a lower user limit was introduced on the Aminet server and the amount of time between two synchronization runs was considerably reduced. Most clients now synchronize their file systems with the server file system three times a day instead of just once a day. Furthermore, the **`new`** directory, where new and unclassified files reside, was made unreadable for users.

Also, the single-directional organization was partly broken up by allowing new files to be submitted to clients. The client then forwards these files to the Aminet server on the next polling run.

6. Conclusions

Distributing archive file systems on several hosts in a wide area network can lead to tremendous savings of network bandwidth and long distance transfers. It makes the sharing of scientific, business and multi-media data much easier by improving availability, access times and fault tolerance and significantly reducing the costs for long-distance file transfers. Facing the exponential growth of wide area networks such as the Internet, the use of distributed file archives is imminent, not only in an academic environment but also for multinational companies.

Several of the problems that need to be solved are, amongst many other factors, the heuristics of the replication process and the consistency between file systems on different sites. While still relying on existing protocols, the *Aminet* project addresses most of the problems of distributed file archives by providing assistance on many levels, from the user-interface, assisting resource discovery, down to the file management software. It serves as a test case for distributed archives with replicating file systems. Although many administration tasks can be automated, a fair amount of manual intervention is still required to maintain the archive as there are many unpredictable factors which decide over the importance or popularity of a file.

The decent experiences gathered during one year of operation of Aminet prove that it is well possible to organize file archives in large scale distributed systems without implementing new protocols. However, new protocols will definitely have to be judged by their amount of support for the organization of file archives in large scale distributed systems.

Acknowledgments

This paper would not have been possible without the help of several fine individuals, especially Silvano Maffeis who assisted and encouraged me in many ways. A very special thanks to my colleagues at the Computer Science Student's Association (ICU) especially Urban D. Müller, who was the driving force behind the Aminet project and who assisted me in gathering statistical data, and Markus Wild for being ready to answer all my Unix-related questions.

References

[1] **Lottor, M. K.** Internet Growth (1981-1991). RFC 1296, SRI International, Jan. 1992.

[2] **Schwartz, M. F. et al.** Supporting Resource Discovery Among Public Internet Archives Using a Spectrum of Information Quality. Tech Rep. CU-CS-487-90, University of Colorado, Boulder, USA, Sept. 1990.

[3] **Ewing, D. J. et. al.** A Measurement Study of Internet File Traffic. Tech. Rep. CU-CS-571-92, University of Colorado, Boulder, USA, Jan. 1992

[4] **Sandhu, H. S. and Zhou S.** Cluster-Based File Replication in Large-Scale Distributed Systems. In ACM Sigmetrics and Performance, Vol. 20, No. 1, June 1992, USA

[5] **Maffeis, S. and Cap C.** Replication Heuristics and Polling Algorithms for Object Replication and a Replicating File Transfer Protocol. Tech. Rep. IFI-92.06, University of Zurich, Dpt. of CS, Aug. 1992

[6] **Maffeis, S.** File Access Patterns in Public FTP Archives and an Index for Locality of Reference. In ACM Sigmetrics Performance Evaluation Review, Vol 20, No. 3, March 1993

[7] **Reynolds, J. and Postel, J.** File Transfer Protocol (FTP). RFC 959, Network Information Center, SRI International, October 1985

[8] **Emtage, A. and Deutsch, P.** archie - An Electromic Directory Service for the Internet. Usenix Conference Proceedings, San Francisco, USA, Jan. 1992

[9] **Cáceres, R**. Measurements of Wide Area Network Traffic, University of California, Computer Science Division, Berkeley, USA, 1989

[10] **Ganatra, N. K**. Census: Collecting Host Information on a Wide Area Network, University of California, Santa Cruz, June 1992

[11] **Horton, M. and Adams, R**. Standard for Interchange of USENET messages. RFC 1036, AT&T Bell Laboratories, USA, December 1987

[12] **Neumann, B. C.** The Prospero File System, Proceedings of the USENIX File System Workshop, Ann Arbor, Michigan, USA, May 1992

[13] **Various,** Gopher, Unix manual page, Univ. of Minnesota, Dept. of Computer Science, Dept. of Computer Sciences, USA, 1992

[14] **Claffy, K. C., Polyzos, G. C., Braun, H.-W.** Traffic Characterstics of the T1 NSFNET Backbone, University of California, Department of Computer Science and Engineering, San Diego, CA, USA, 1992

[15] **NSFNET Service Center**, NSFNET traffic statistics, available by anonymous ftp from nis.nsf.net in `/statistics/NSFNET`, February 1993

[16] **McLoughlin, L**. Mirror - Mirror packages on remote sites, Unix manual page, August 1991

[17] **Wall, L**. Perl - Practical Extraction and Report Language, Unix manual page, NASA Jet Propulsion Laboratory, USA, October 1989

[18] **Wall, L. and Schwartz, R. L.** Programming perl, A nutshell handbook, O'Reilly Sebastopol, USA, 1991

[19] **Danzig, B., Hall, R. S., Schwartz, M. F.** A Case for Caching File Objects Inside Internetworks, Tech. Rep. CU-CS-642-93 University of Colorado, Boulder, USA, March 1993

[20] **Kotler, P.** Globalization - Realities and Strategies, in «Die Unternehmung» 2/90, Verlag Paul Haupt Bern, Switzerland, 1990