LaRose, Sharon Lewis, Keith Moore, Andrew Pearson, Jon Richardson, Bill Rosener, and Andrea Van Hull all made valuable contributions to the design, development, and testing of various versions of `xnetlib`. We also thank the many users of early versions of `xnetlib` for their constructive comments and for their patience.

References

[1] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. GAMS - A framework for the management of scientific software. *ACM Transactions on Mathematical Software*, 11(4):313–355, December 1985.

[2] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshamn. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

[3] J. Dongarra and E. Grosse. Distribution of mathematical software via electronic mail. *Communications of the ACM*, 30(5):403–407, May 1987.

[4] J. Dongarra and B. Rosener. NA-NET numerical analysis net. Technical Report ORNL/TM-11986, Oak Ridge National Laboratory, Oak Ridge, TN, December 1991.

[5] S. Feldman, D. Gay, M. Maimone, and N. Schryer. A Fortran-to-C converter. Computer Science Technical Report 149, AT & T Bell Laboratories, Murray Hill, NJ, May 1990.

[6] M. Kent. The numerical analysis net. Technical Report 85, Eidgenössische Technische Hochschle (ETH), Zurich, January 1988.

[7] A.A. Pollicini, editor. *Using Toolpack Software Tools*. Kluwer Academic, 1989.

return the output.

Another proposed capability will allow users to add their own servers to the collection of servers globally available through `xnetlib`. Any `xnetlib` user could then access this contributed software by adding the appropriate server to his active server list. This feature will greatly expand the amount of software available through `xnetlib`.

We also plan to expand the scope of the `xnetlib` system. In the future `xnetlib`'s distributed repository will include more non-mathematical software and more reports. Additional software and document collections will be linked into `xnetlib`'s existing collection.

Starting with LAPACK, `xnetlib` began distributing entire libraries. This service will be extended to other libraries.

`Xnetlib` already provides fast and easy access to a large collection of mathematical software. In the future `xnetlib` will provide greatly expanded capabilities and will be much closer to being a complete problem solving environment.

6 Summary

Software distributed by `netlib` comes with the disclaimer that “anything free comes with no guarantee”. In contrast to commercial vendors like NAG and IMSL, `netlib` offers no support beyond whatever documentation contributing authors choose to provide with their code. These caveats also apply to `xnetlib`.

On the other hand, both `netlib` and `xnetlib` provide free, easy access to a large body of high-quality code, and the phenomenal growth of `netlib` over the past eight years attests to the value of this service. We hope that `xnetlib`, by making this high-quality code even more accessible, will encourage software developers to make their codes freely available and will make good programming easier for the scientific computing community.

Acknowledgements

`Xnetlib` is the product of the efforts of many people. Ken Bateman, David Bolt, Shirley Browne, Jennifer Finger, Tracy Gangwer, Stan Green, Brian

ical analysis and other disciplines. For simple searches the user need only enter an individual's last name. The **modify search** feature can be used for more elaborate searches and the **modify listing** feature can be used for controlling the form of the output.

4 Getting started

`Xnetlib` requires release 4 (or later) of X11 and the Athena widget libraries as supplied by MIT. The executable for the `xnetlib` client requires approximately 200 kilobytes on a Sun SPARCstation2. The locally cached indexes occupy zero to 850 kilobytes depending on how many index files are cached.

The first step in installing `xnetlib` is to obtain the source code for the `xnetlib` client. The software for the `xnetlib` client is itself available from both `xnetlib` and `netlib`, so a simple way to obtain the source is to send the message **send xnetlib.shar from xnetlib** to `netlib@ornl.gov`. `Netlib` will respond by sending a `shar` file containing all necessary source code and documentation. `Xnetlib` is also available by anonymous ftp from `cs.utk.edu` in the `pub/xnetlib` directory. The `xnetlib` distribution includes an Imakefile so installation is normally trivial if the X Window System has been configured properly on the client machine.

`Xnetlib` is easily customized. One common customization at multi-user sites is to have a single cache of index files so indexing information can be shared by all local users.

5 Plans

Plans are already under way to expand `xnetlib`. One major addition will be the capability of remote execution. Many useful utility programs are large, making distribution tedious, or are more expensive to build than to execute. In such cases, allowing remote execution may be a better use of resources from both the distributor's and user's point of view. `Toolpack` [7], a large collection of Fortran software tools, and `f2c` [5], a Fortran-to-C translator, are logical candidates for this remote execution service. Users, instead of downloading, installing, and executing these programs, could submit suitable input and have a machine at a remote server site execute the programs and statistical analysis to find useful matches that may not be uncovered by other types of searches. In contrast, the fuzzy search capability in `WAIS` is based on an heuristic rather than statistical approach.

Clicking on the **download** button causes `xnetlib` to display a list of selected software and documents. Clicking on the **download path** button allows a user to change the directory to which files will be downloaded. The **dependency checking** button is a modal switch. If **dependency checking** is off `xnetlib` will send only the selected routines. If it is on, `xnetlib` will send the selected routines and any routines that they call. If the user is satisfied with the selection list and the target directory, he should click on **get files now** to initiate the transfer. Figure 5 shows the downloading of selections made from the LAPACK, SOFTLIB, and RN libraries. These three libraries reside in repositories at separate sites. The selected files will

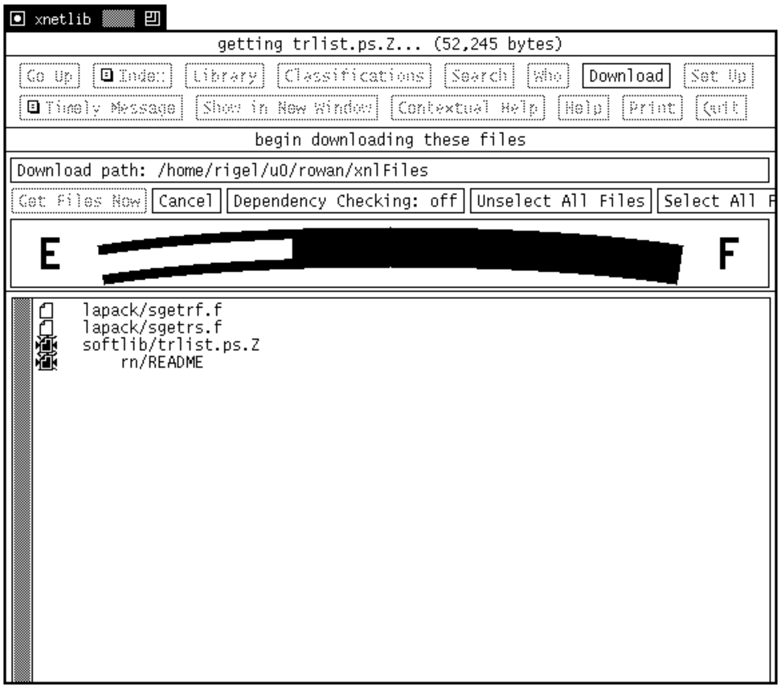

Figure 5: Downloading the selected routines.

appear in the specified directory, usually within a few seconds.

The **who** command allows a user to search the NA-NET White-pages [4, 6], a database containing information about individuals interested in numer-

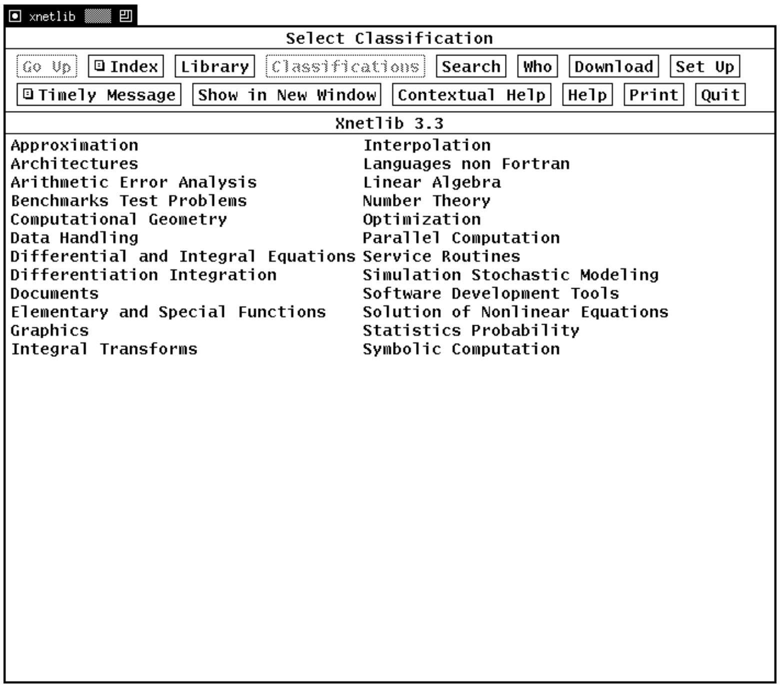

Figure 4: Classification menu.

The complete contents list of LAPACK is too large to fit in the window, but

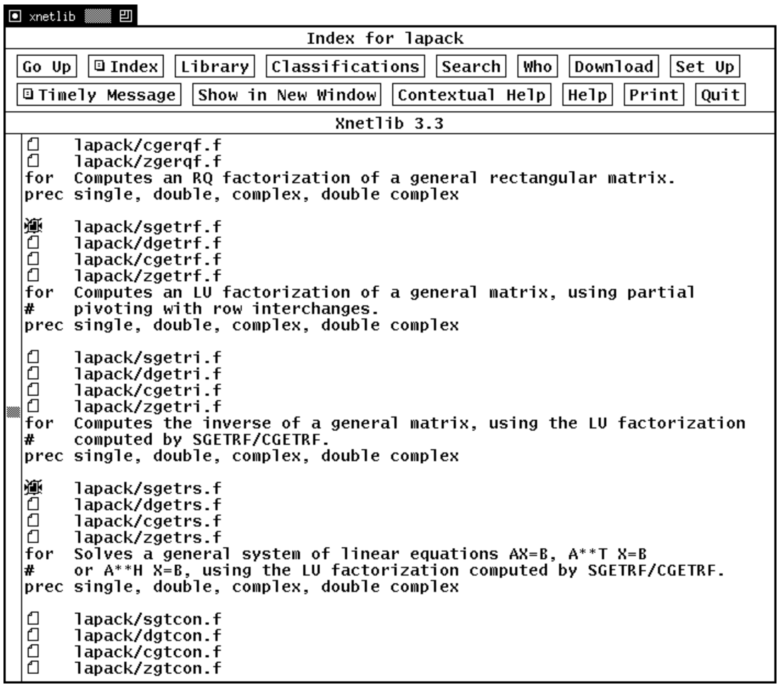

Figure 3: LAPACK index.

the provided scrollbar allows the user to scroll through the rest of the list. Clicking on the box adjacent to a routine name selects that routine for future downloading. In Figure 3 the user has selected **sgetrf** and **sgetrs** from the LAPACK library.

The **classification** feature allows a user to narrow a search. The classification of the `xnetlib` software libraries is based on the GAMS [1] classification system, augmented to include classifications other than mathematical software. Selecting **linear algebra** causes the names of the libraries with linear algebra software to be displayed (Figure 4).

A user may also wish to **search** by keyword instead of viewing the contents of a particular library. In this mode, descriptions of files are searched by a keyword string the user provides. The keyword search can be a search on the intersection or union of the words in a search string, a literal search for an exact string (with or without case sensitivity), or a fuzzy search based on the latent semantic indexing technique [2]. Latent semantic indexing uses

Many Internet sites have sizable collections of documents or software. It is both unnecessary and undesirable to require that these collections reside at a single site. `Xnetlib` gives users access to a distributed repository of software and documents by establishing socket-based links with the repository sites. Users have access to any or all of these repositories through a single interface.

`Xnetlib` users control which sites are linked into the distributed repository using the **set up** button. Current repository sites include `netlib@ornl.gov`, `spark.brl.mil`, and `softlib@rice.edu`. Clicking on the **timely message** button displays news about individual repository sites and clicking on the **index** button displays their general indexes.

Clicking on the **library** button shows the libraries that are available through `xnetlib`. Figure 2 shows a unified list of software and documents available at the repositories of Oak Ridge National Laboratory, Rice University, and the U. S. Army Research Laboratory.

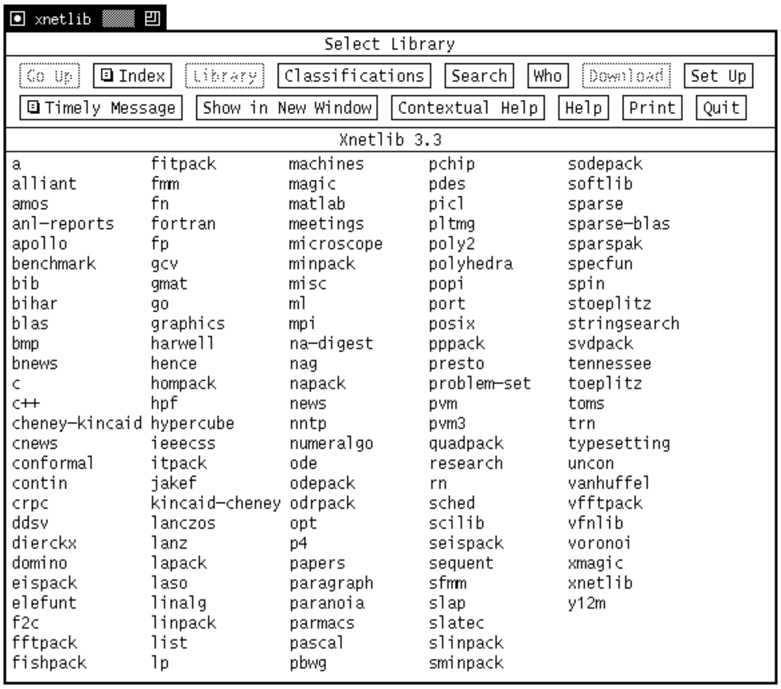

Figure 2: Library menu.

Clicking on a library name lists the contents of that library. For example, clicking on **lapack** displays a partial listing of LAPACK's contents (Figure 3).

tem consists of the `xnetlib` server process running on machines at the repository sites, `xnetlib` X client processes running on users' local machines, and TCP/IP socket-based communication links between these clients and servers.

`Xnetlib`'s server process runs continually at an `xnetlib` repository site, listening for incoming requests from `xnetlib` client processes. Typically, the `xnetlib` server runs on the same machine and accesses the same software repository as the `netlib` server. Upon receiving a request, the `xnetlib` server determines the nature of the request and responds by transferring the appropriate file from the repository to the `xnetlib` client process.

The `xnetlib` client running on the user's local machine provides an X Window interface to the `xnetlib` repository. It is programmed in C, using the Athena widget libraries. This interface makes searching through the software and document collection easy. For example, an `xnetlib` user can view the contents of any library simply by clicking a button. Other commands, such as keyword searching or requesting software, also require just a few button clicks. To avoid unnecessary communication between the client and server, requested indexes are cached locally. Frequently requested information can therefore be quickly retrieved from local cache instead of repeatedly retrieved from the remote server. Other requests are passed to the server via sockets. Section 3 describes the use of the interface in more detail.

3 Features

`Xnetlib` features and capabilities include:

- Access to a distributed repository
- Searching by a software libraries list
- Searching by software classifications
- Searching by keyword
- Software and document retrieval
- Access to the NA-NET White-pages
- Online help

Record keeping. The system should have the capability of logging requests so updates and corrections can be reported to users.

Security. The system should be secure from accidental or intentional misuse.

Portability. The system's implementation should be as portable as possible.

Accessibility. The system should be accessible to a large number of users.

No existing software and document retrieval system satisfied these requirements. `Ftp` and `archie` are fast and portable but are not sufficiently flexible in their searching or record keeping capabilities. In addition, in `archie` the indexing mechanism and the large volume of accessible material make fully up-to-date indexing information impractical. `Gopher` is geared to browsing through the Internet rather than to retrieval of materials, while `WAIS` is better suited for retrieval of documents than for retrieval of software.

`Archie`, `gopher`, and `WAIS` have different goals than `xnetlib` and should not be regarded as competitors. Their use can, in fact, be complementary. For example, `gopher` can be used to provide improved accessibility to `netlib` and `xnetlib`.

Figure 1 shows the basic configuration of the `xnetlib` system. The sys-

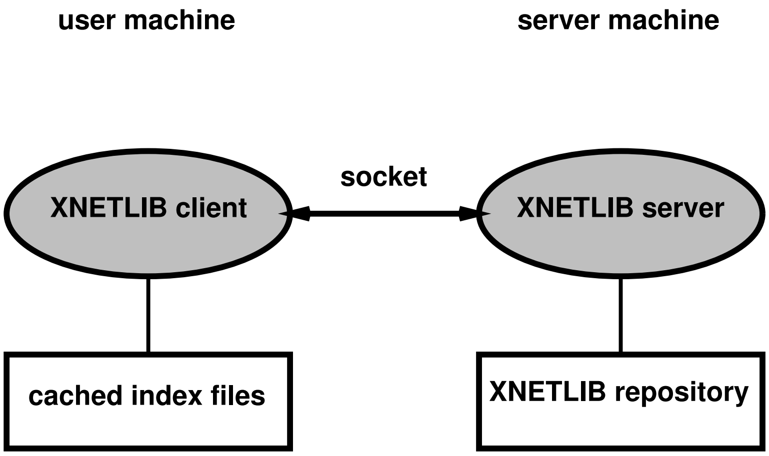

Figure 1: `Xnetlib` configuration.

users to search through a large distributed collection of software easily and to retrieve requested software in seconds.

1 Background

`Xnetlib`'s predecessor, `netlib`, grew from a need to have a quick and easy method for distributing small pieces of mathematical software. `Netlib` services began in 1985 at two sites, Argonne National Laboratory and AT&T Bell Laboratories, and distributed software from about 30 libraries. For additional information about `netlib`'s operation and use see the introductory paper by Dongarra and Grosse [3].

One of the changes since `netlib`'s introduction has been the transfer of `netlib` services from Argonne National Laboratory to Oak Ridge National Laboratory. Also, the availability on `netlib` of the `netlib` program itself has enabled many other sites to set up their own software repositories.[2] The `netlib` software collection has now grown to well over 100 libraries. The number of software requests sent to `netlib` also has grown dramatically. The most heavily used `netlib` server, at Oak Ridge National Laboratory, processed over 150,000 requests last year.

2 Overview

We based `xnetlib`'s design on the following requirements:

Speed. Retrieving software should take seconds, not minutes as typically required by e-mail.

Usability. The user interface should make searching through a large collection of software and documents easy.

Organization. The system's repository should be a moderated collection, with up-to-date indexes, and a database organized to facilitate searching and ease of retrieval. The repository may be distributed over several sites.

[2]Send the message **send index** or **send sites from netlib** to `netlib@ornl.gov` to receive a list of `netlib` sites.

Software Distribution Using XNETLIB

*

Jack Dongarra[†] Tom Rowan[‡] Reed Wade[§]

July 29, 1993

Abstract

`Xnetlib` is a new tool for software distribution. Whereas its predecessor `netlib` uses e-mail as the user interface to its large collection of public-domain mathematical software, `xnetlib` uses an X Window interface and socket-based communication. `Xnetlib` makes it easy to search through a large distributed collection of software and to retrieve requested software in seconds.

`Xnetlib` is a new software distribution tool recently developed at the University of Tennessee and Oak Ridge National Laboratory. The goal in developing `xnetlib` was to provide Internet users faster and easier access to `netlib`'s large collection of software, data, and documents. Unlike `netlib`, which uses e-mail to process requests for software, `xnetlib` uses an X Window interface and socket-based communication between the user's machine and the `xnetlib` server[1] machine to process software requests. This enables

*This work was supported in part by DARPA and ARO under contract DAA 03-91-C-0047, and in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract DE-AC05-84OR21400.

[†]University of Tennessee and Oak Ridge National Laboratory (`na.dongarra@na-net.ornl.gov`)

[‡]University of Tennessee and Oak Ridge National Laboratory (`na.rowan@na-net.ornl.gov`)

[§]University of Tennessee (`wade@cs.utk.edu`)

[1]Throughout this paper, *server* refers to the process handling software requests and not to the X display server.

CS - 93 - 191

Software Distribution Using XNETLIB

Jack J. Dongarra, Tom Rowan and Reed Wade

Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301

and

Mathematical Sciences Section
Oak Ridge National Laboratory
Oak Ridge, TN 37831

CS - 93 - 191