Workstations and the Virtual System Model

B. Clifford Neuman

Department of Computer Science and Engineering
University of Washington, FR-35
Seattle, Washington 98195

bcn@cs.washington.edu

Technical Report 89-10-10
October, 1989

1 Introduction

The role of workstations in future systems has been a hotly debated topic. Some believe that the workstation is useful primarily as a terminal and that the computing of the future will be done on large multiprocessor systems. Others believe that workstations *will be* the processors of those multiprocessor systems. Still others believe that the role of the workstation will be between the two extremes, providing simple processing capabilities such as editing for the local user.

One thing that is clear is that users of the future will require access to objects and services that are scattered across large networks. A clear role for the workstation is to forge these objects and services into complete systems. In order to talk about the role of the workstation in such systems we must have an idea of how these systems will be organized. This report briefly describes one approach.

2 The Virtual System Model

As the size of distributed systems has grown it has become increasingly difficult for the user to organize and find objects and services. Today's users are able to cope because only a tiny portion of the objects and services in the world are available to them. Users needn't be concerned with objects in which they have little interest. One way to help users cope with very large systems is to let each user see a smaller system containing only those parts that are of interest.

This section briefly describes the *virtual system model*[6], a new model for organizing large distributed systems. The virtual system model supports the maintenance of customized views of the system. Each view is called a *virtual system*.

The virtual system model addresses several of the problems that arise in large systems. By allowing customized views, the user only sees the parts of the system that are of interest. Objects can be organized in multiple ways with objects appearing at multiple points in multiple virtual systems.

This research was supported in part by the National Science Foundation under Grants No. DCR-8420945 and CCR-8611390.

This makes it easier to find objects since one can look for an object using whichever organization is most appropriate based on the information available. The virtual system model supports the choice of attributes and the selection of servers when a particular service is needed. It also plays a role in the selection of processors on which programs are to be run, and on the authentication and security mechanisms to be used when a user's objects are accessed. Finally, a virtual system can be bound to a program and can be used to specify the environment in which the program runs.

2.1 The Prospero File System

Prospero is a distributed operating system that is based on the virtual system model and that is presently under construction at the University of Washington. Because of the large number of files on even small collections of systems, the file system provided a natural example and the focus for a prototype implementation to test the ideas of the model. The prototype is presently running at the University of Washington and on several sites in another part of the Internet.

The global file system is a collection of *virtual file systems*, each associated with one or more virtual systems. Although each virtual file system has a root, its directories are not required to be organized as a tree. Cycles are allowed and objects and directories can appear in multiple file systems, or even with different names in the same file system. Directories in Prospero may be connected by links which change the way the target directory is viewed. The directories, files, and the links connecting them form a generalized directed graph. Virtual file systems grow as users learn of objects or directories outside their current environment and add them to their own virtual file systems.

The principal component of the Prospero file system is the directory service. Directories map from names to object pointers. In the file system the objects are files and directories. There is no requirement that the objects included in a directory be located at the same storage site. Together with the ability to reference an object in multiple directories, this allows one to easily organize objects in multiple ways. For example, papers can be organized by author, or subject, yet only one copy of each paper must be maintained.

A virtual system starts out as a copy of a prototype. Initially, the virtual system might include references to indices by topic, and author. References to other well known virtual systems might also be included. As the user finds objects or hierarchies of interest through communication with others, by exploring parts of the prototype virtual system, or through other means, these can be added to the virtual system. It is expected that most virtual systems will include objects which the user does not know about, but these objects may appear deep in the hierarchy. If some of these pieces are found to be of interest, it is expected that the user would explicitly include them closer to the root.

Users are able to tailor their virtual systems at several levels. They can decide for themselves which hierarchies, indices, or other organizational mechanisms are to be included in the root, or lower levels of their virtual systems. When a hierarchy is included, it can be included as a subdirectory, or its contents can be merged with other objects at the present level. Finally, it is possible to include programs or filters on links so that only those objects in the included hierarchy that match a set of criteria are included in the filtered view of the hierarchy. These programs can also add objects to that view and alter the names or attributes of included objects.

A customizable name space has the potential to cause confusion and to limit sharing. To address this problem, Prospero supports closure[5]. A namespace is associated with every object. In this way, the context within which a name is to be resolved is automatically passed along with the object specifying the name. Although the same name may refer to different objects within different contexts, the correct context is always known.

2.2 Comparison to Other Systems

Unlike symbolic links in systems such as NFS[7], Andrew[3], and Locus[8], all names for an object in Prospero are equivalent. In NFS, Andrew, and Locus, each object has a primary name, and all other names are translated to the primary name before the object is accessed. In Prospero, all names for the object are primary. Further, an object with any name can physically reside at any location. In NFS, Andrew and Locus files are distributed at the file system or volume level. All files whose names begin with a particular prefix must physically reside on the same disk. The fact that this is not required in Prospero makes it much easier to place files where they are most frequently accessed with no need to move an entire hierarchy.

The Prospero file system is similar to QuickSilver[1] and Tilde[2] in that it supports user centered naming. Each user chooses the parts of the system that will be visible, and this choice follows the user around regardless of the physical processor used. A key difference is that the Prospero file system is only one example of a service based on the virtual system model. The virtual system model extends this customizability beyond the file system. Authentication, authorization, and the selection of services and processors are all affected by the active virtual system.

The Prospero file system differs from QuickSilver and Tilde in several additional ways. When using Prospero, it is possible to access objects in other virtual systems directly. QuickSilver requires that the object first be be added to the user's system. Tilde requires that the entire hierarchy containing the desired object be added to the user's Tilde forest. Both QuickSilver and Prospero allow links to individual objects or subdirectories. Tilde does not. In Prospero all names for an object are equivalent. Although QuickSilver supports several varieties of links, it is not possible to have a hard link to an object stored at a storage site different than that of the enclosing directory. Finally, through union links and filters, Prospero allows customization of included hierarchies at a finer level of detail than either QuickSilver or Tilde.

Because the directory service can be used to name objects other than files and directories, and because the objects included in a directory do not need to be located on the same system as the directory itself, the directory service is actually much closer to directory services in object-based systems such as Amoeba[4] than to directories in traditional file systems. The Prospero directory service differs from that found in capability based systems in that no permission is implied by one's ability to name an object.

3 Role of the Workstation

A workstation is its user's window to the world. Because it is situated between the user and the network, it is in the ideal position to maintain the user's view. The workstation can make the necessary queries to translate names in the user's virtual system to the information actually needed to access the object. If programs are to be run, they can either be run locally, or started on a remote processor. This decision will be based on information that is part of the user's virtual system, or perhaps in the virtual system attached to the program to be run.

In this role, the workstation acts as an agent for the user. The user's environment and shell would be maintained on the workstation and the workstation would decide where each command is to be executed. Once the decision has been made, the workstation would then contact the appropriate server and cause the command or program to be run. It is only when a remotely executed program does terminal I/O that the user directly interacts with a remote system.

The virtual system model does not prevent the workstation from playing a more active role in a large distributed system. Files can be stored and services provided by workstations, but this is not required. The virtual system model sees the workstation as equivalent to all other nodes in the system and will let the workstation provide such services if desired. For availability reasons, however, one might only want to do this in certain cases.

4 Conclusions

As the size of distributed systems grow it becomes increasingly difficult to keep track of all the information and services that are available. One way to deal with this problem is to allow the user to tailor the view of the system so that only those parts that are of interest are immediately visible.

The workstation is the user's interface to the system. Because of its location relative to the user, it is the right place to maintain the user's view. The workstation should act as the user's agent in communicating with and requesting services from the external world.

Acknowledgements

Ed Lazowska, Hank Levy, David Notkin, John Zahorjan, and Brian Bershad helped me to refine the ideas presented in this report. Ed Lazowska deserves special thanks for the numerous discussions out of which many of the ideas emerged. Some of the ideas emerged from discussions with Alfred Spector and other instructors at the Arctic '88 Course on Distributed Systems.

References

[1] Luis-Felipe Cabrera and Jim Wyllie. QuickSilver distributed file services: An architecture for horizontal growth. In *Proceedings of the 2nd IEEE Conference on Computer Workstations*, pages 23–27, March 1988. Also IBM Research Report RJ 5578, April 1987.

[2] Douglas Comer and Thomas P. Murtagh. The Tilde file naming scheme. In *Proceedings of the 6th International Conference on Distributed Computing Systems*, pages 509–514, May 1986.

[3] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in a distributed file system. *Transactions on Computer Systems*, 6(1):51–81, February 1988.

[4] S. J. Mullender and A. S. Tanenbaum. The design of a capability-based distributed operating system. *The Computer Journal*, 29(4):289–299, 1986.

[5] B. Clifford Neuman. The need for closure in large distributed systems. *Operating Systems Review*, 23(4):28–30, October 1989.

[6] B. Clifford Neuman. The Virtual System Model for large distributed operating systems. Technical Report 89-01-07, Department of Computer Science, University of Washington, April 1989.

[7] R. Sandberg et al. Design and implementation of the Sun network file system. In *Proceedings of the Summer 1985 Usenix Conference*, pages 119–130, June 1985.

[8] B. Walker, G. Popek, R. English, C. Kline, and G. Thiel. The Locus distributed operating system. In *Proceedings of the 9th Symposium on Operating Systems Principles*, pages 49–70, October 1983.