The Need for Closure in Large Distributed Systems

B. Clifford Neuman

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

bcn@cs.washington.edu

Abstract

As the size of a distributed system grows it becomes increasingly difficult for users to find their way around. One solution for this problem is user centered naming. A problem with user centered naming is that it has the potential to cause confusion and to make sharing difficult. This problem can be avoided if the system supports closure. Although the same name may refer to different objects at different times, the context in which the name is to be evaluated is always known.

1 Identifying Objects in Very Large Systems

As distributed systems become larger it becomes increasingly difficult to find things. Traditional system such as Andrew[3], Locus[7] and Sprite[6] support a uniform global namespace which uniquely names all objects. While this approach allows simple sharing of names, the names tend to be lengthy. Another problem with this approach is that, to the user, the namespace is huge and mostly cluttered with things that aren't of interest. This makes the job of finding things more difficult.

An alternative to a uniform namespace is to allow individual users to customize their namespace so that they only see the objects that are of interest. This approach is taken in several recent distributed systems. Among them are Tilde[2], QuickSilver[1] and Prospero[5]. An objection to user centered naming is that the same name can refer to different objects when used by different users. It is claimed that this can lead to confusion and that it limits sharing.

It should be pointed out that even systems supporting a uniform global namespace allow some degree of customization. Users often name objects relative to local directories, and environment variables set in initialization scripts control the directories searched for programs, libraries, and other files. This has the potential to be more confusing than the user centered namespace because the customization is not explicit. For example, consider trying to compile a program written by another user. If the program depends on libraries or include files found in the user's search path, it might not compile for a different user.

This research was supported in part by the National Science Foundation (Grants No. DCR-8420945 and CCR-8611390), US WEST Advanced Technologies, and Digital Equipment Corporation.

B. Clifford Neuman. The Need for Closure in Large Distributed Systems. *Operating Systems Review.* 23(4):28-30, October 1989.

What is needed is a mechanism that allows customization, and makes it explicit. Further, the mechanism that is chosen must not make it difficult to resolve names that have been specified by others.

2 How Closure Solves These Problems

It is possible to get the best of both worlds by drawing on the concept of closure as it is applied in lexically scoped programming languages. A closure is a combination of code to be executed, and the context within which variables are to be evaluated. Our belief is that a namespace should be associated with every object in a distributed system. When running a program, names can be resolved in the appropriate namespace. By default, names hardcoded into the program would be resolved in the namespace associated with the program, those specified by the user would be resolved in the user's namespace, and those included in the file being processed would be resolved in the namespace attached to that file. In this way, the context within which a name is to be resolved is automatically passed along with the object specifying the name.

By default, closures should be dynamic. This allows changes in a namespace to be visible to the objects attached to that namespace. There are instances, however, where a closure should be frozen. This is important for security. Without it, one has little control over the source or destination of data flowing into and out of a program through a file named in the closure. Such control is necessary if a program is to be confined[4][1].

In order to support closure in the manner described it must be possible to succinctly and automatically specify the namespace that is attached to each object. Prospero supports such a specification.

3 Support for Closure in Prospero

Prospero is a distributed operating system that is based on the virtual system model[5] and that is presently under construction at the University of Washington. It supports a user centered view of the objects and services available on a network. Each view is called a virtual system. Tools are provided to help users customize and organize their views of the world. Among these tools are filters and union links[2].

A virtual system is a namespace. Each is identified by a name. In order to form a closure, only the name must be stored along with the attached object. This succinct representation makes support for closure possible. An attached virtual system is one of the many file attributes that is supported by Prospero.

Support for closure is being added to Prospero. Once a virtual system has been associated with

[1] A program is confined when data is not leaked (placed in locations not intended by the user).

[2] A filter is a program that can modify the results of a directory query where the path for the queried directory passes through the filtered link. A union link allows the results of a (possibly filtered) query of the linked directory to be merged with the contents of the directory containing the link.

each object, all that is needed is a mechanism that will, for each process, keep track of the virtual systems associated with the user, the running program, and all open files. The name resolution mechanism can then be modified to allow the programmer to specify the virtual systems to be searched when resolving a name.

4 Conclusions

As systems become larger, it becomes harder for users to find things. User centered naming can help the user deal with scale, but it also has the potential to cause confusion and to make sharing difficult. This problem can be addressed by supporting closure. Although the same name may refer to different objects at different times, the context in which the name is to be evaluated is always known. Support for closure is being added to the Prospero distributed system. Only experience will tell how useful the concept of closure is in such systems.

Acknowledgements

I would like to thank Ed Lazowska, Brian Bershad, Robert Henry, David Notkin and John Zahorjan for discussions and comments on earlier drafts of this paper.

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 1, 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] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, October 1973.

[5] 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.

[6] John K. Ousterhout, Andrew R. Cherenson, Frederick Douglis, Michael N. Nelson, and Brent B. Welch. The Sprite network operating system. *Computer*, 21(2):23–35, February 1988.

[7] G. Popek and B. Walker, editors. *The Locus Distributed System Architecture*. M.I.T. Press, Cambridge, Massachusetts, 1985.