MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Using Reconciliation to Share Files between Occasionally Connected Computers

John H. Howard

TR93-08a December 1996

Abstract

Future large distributed systems will be made by interconnecting highly autonomous subsystems, rather than by building ever more elaborate complexes which attempt to provide a single system image transparent to the user. The work described here explores the implications of this in the context of file sharing using occasional reconciliation.

Fourth Workshop on Workstation Operating Systems (WWOS-IV), Napa, California, October 14-15, 1993

Revision History:—

1. First version: May 27, 1993
2. Reformatted: July 7, 1994

Introduction

Most distributed file systems attempt to provide the impression of a single, globally consistent system. This simplifies things for users, avoiding the need to keep track of multiple servers; and for operators, making it possible to reconfigure servers, migrate files, and perform all sorts of maintenance without interfering with users or requiring their cooperation. Furthermore, it allows application programs, written to deal with a single system, to continue to be used unchanged in a distributed environment.

There are several arguments against the use of single system images, however. They require frequent communication between the various servers and clients, often depending on high performance networks. Therefore, they are unsuitable for mobile computers. They usually go to great lengths to provide complete consistency, regardless of whether it is really needed or wanted. Immediate and automatic distribution of new files and versions of files may not be what the user wants, as it reduces user control over data sharing, and poses security risks. Sophisticated users and application programs may prefer to take advantage of desirable features of the distributed environment. Since application programs can not be expected to help at all dealing with network failures, transparent systems must go to great lengths to handle them completely.

Recent work in file systems for mobile computing addresses the problem of lost or slow communication by allowing some form of disconnected operation. However, the underlying philosophy is still based on a single system image; the disconnected client is considered to be in a sort of error condition that should be rectified as soon as the file server connection is restored.

The work described here takes a different philosophical position, namely, that the file system is built out of separate, autonomous parts that communicate occasionally to reconcile their different versions of the files they "share". There is no single authoritative source, but rather a collection of peers. Reconciliations happen at times convenient for the user or application, rather than continuously. Any participating system can choose to operate completely independently and autonomously for an indefinite period.

This approach has its advantages and disadvantages. Advantages include independence from network connectivity and performance, easy mobility, and the ability to make and exploit private versions of shared information. Disadvantages include the need to specify reconciliation times and to deal with conflicts between inconsistent versions. It is the purpose of the work in progress reported here to explore these advantages and disadvantages by means of a prototype.

Work in progress

The heart of the experimental system being developed is a program, RECONCILE™, which compares and reconciles selected file system hierarchies. Briefly, it builds a journal for each site (that is, the file system sub-tree being reconciled) by comparing the file system state as described in the pre-existing journal with the actual state observed when the program runs. The journals record file system activity such as file creations, updates, and deletions. The program compares the journals in order to identify and propagate updates to sites that lack them. Conflicts arising from independent updates at multiple sites are detected and reported reliably. The journals also contain information about other sites and the last time each was contacted; this information is used to determine when it is safe to discard obsolete journal entries.

Current versions of the RECONCILE program exist for DOS and UNIX platforms, and a Macintosh version is under development. There are a number of specific applications being explored:

- Transporting files between personal computers at different locations (office and home, or desktop and portable) by way of diskettes (this is often referred to as "sneakernet"). This involves reconciliations between three "sites": the two computers and the diskette that is carried back and forth, to be reconciled with the working directory of whichever personal computer is being used at both the beginning and the end of each working session. If a reconciliation is forgotten or the disk left behind, it is possible to continue work using the existing versions. The RECONCILE program will detect and report any conflicts, which the user must then resolve by hand. Personal experience to date suggests that this is relatively infrequent and not too to handle.

- Transport between different platforms (DOS and UNIX) for software development. This is similar to the sneakernet application above, with the added feature that the program automatically recognizes platform types and adjusts text file formats accordingly.

- Joint development in a shared project. Each developer has a private copy of the entire project, and there is a shared copy accessible to all. Reconciliations (possibly constrained to copy files only in one direction) are used to check working copies in and out and to report conflicts, which must be resolved manually. Intermediate reconciliations may be used at appropriate moments to reduce the impact of accumulated inconsistencies.

- New release installation. In this case the sites are the directories on workstations used to store program products and the distribution directories containing the new releases. Installation uses one-way reconciliations, thus allowing for the preservation of local customizations and detection of conflicts between them and the new releases. It is expected that this approach would work well for subsystems and programs that use a

---

™ RECONCILE is a trademark of Mitsubishi Electric Research Laboratories, Inc.

single installation directory (or hierarchy), it would not handle programs that require installation of files in a number of separate system directories.

Note that transparency and a single system image is inappropriate in all of these applications. In every case, the user is directly aware of the existence of multiple sites and versions and sees as being an advantage rather than an inconvenience.

Other applications could also be imagined. For example, the reconciliation program could be extended to merge individual records in files containing personal databases such as phone lists or calendars and then used to maintain and share the databases as above. For such cases, the application program might do occasional reconciliations automatically, leading to a partially transparent system in which temporary inconsistencies are tolerated but expected to be infrequent.

Related Work

Distributed file systems that support disconnected operation include Coda [Kistler 92] and Ficus [Guy 91]. Both of them use optimistic concurrency control based on time stamps to detect conflicting updates during reintegration, but both retain the notion of a single system image rather than exposing multiple versions to users and application programs.

Several commercial program products such as LapLink V [Laplink 93] and FileRunner [Filerunner 93] provide a limited form of reconciliation based on copying newer files. Simple time-based directory merging (used in some copy commands and in LapLink) does not detect conflicting updates, and responds to deletions by restoring the deleted file from the other site rather than by propagating the deletion action. FileRunner handles deletions and detects some multiple update conflicts, but is organized around one master site and a number of secondary ones.

The lazy replication system of Ladin, Liskov, Shira, and Ghemawat [Ladin 92] uses time stamps to decide to propagate recent events in a general object system. Systems such as this recognize multiple versions internally but do not make them visible to the user or application program.

Grapevine [Grapevine 82] provided mail and user directories across a large organization; it was able to deal with possible inconsistent versions because of the specific application it supported.

There is a large body of work such as ISIS [Birman 87], distributed databases, and general distributed object systems [Herlihy 90] which allow applications to choose levels of consistency for their operations in one way or another. Generally speaking, the weaker consistency levels relax the order at which operations might be observed [Eswaren 76, Lamport 78], but do not work in terms of multiple sites or versions of the same object.

References

[Birman 87] Birman, K. and Joseph, T., "Reliable communication in the presence of failures", *ACM Trans on Computing Systems 5*, 1 (Feb 87), 47-76.

[Eswaren 76] Eswaren, K., Gray, J., Lorie, R., Traiger, I., "The notions of consistency and predicate locks in a database system", *Comm ACM 19*, 11 (Nov 76), 624-633.

[Filerunner 93] *FileRunner*, MBS Technologies, Inc., 4017 Washington Rd #4000, McMurray, PA 15317.

[Grapevine 82] Birrell, A., Levin, R., Needham, R., and Schroeder, M., "Grapevine: An Exercise in Distributed Computing", *Comm ACM 25*, 4 (April 1982), 260-274.

[Guy 91] Guy, R., *Ficus: A Very Large Scale Reliable Distributed File System*, (Ph.D. thesis, June 1991), UCLA Computer Science Department technical report CSD-910018.

[Herlihy 90] Herlihy, M. and Wing, J., "Linearizability: A Correctness Condition for Concurrent Objects", *ACM Trans on Programming Languages and Systems 12*, 3 (July 1990), 463-492.

[Kistler 92] Kistler, J., and Satyanarayanan, M., "Disconnected Operation in the Code File System", *ACM Trans on Computer Systems 10*, 1 (Feb 1992), 3-25.

[Ladin 92] Ladin, R., Liskov, B., Shrira, L., and Ghemawat, S., "Providing High Availability Using Lazy Replication", *ACM Trans on Computer Systems 10*, 4 (Nov 1992), 360.

[Lamport 78] Lamport, L., "Time, Clocks, and the Ordering of Events in a Distributed System", *Comm ACM 21*, 7 (July 1978), 558-565.

[Laplink 93] *Laplink IV*, Traveling Software, Inc., 18702 North Creek Parkway, Bothell, WA 98011.