C0PE: Consistent 0-Administration Personal Environment

Praveen Yalagandula Lorenzo Alvisi Mike Dahlin Harrick Vin

Department of Computer Sciences
The University of Texas at Austin

1 Introduction

The C0PE project explores system issues that arise when an individual's computing environment is spread across dozens of information devices. The goals are (i) to provide a consistent view of each individual's personal information space, (ii) to drive the incremental cost of adding, maintaining, and using additional devices near zero, and (iii) to provide a highly trustworthy environment.

Because of the remarkable successes of hardware engineering and architecture as well as economies of scale, powerful $100 and even $10 information access devices are on the horizon. In the future, users will each have dozens of devices to access data and services. These devices include both traditional computing and communication devices (e.g., desktop computer, palmtop computer, cell phone) and sensors and actuator devices (e.g., home security camera, digital still camera, VCR, front door lock).

This environment raises several challenges.

- Users interacting with an information system through many different devices should perceive a single, consistent view of the underlying data and services. A user should be able to read any file from any device without worrying about where the most recent update was made. In addition, a user should be able to access the "sensors" and "actuators" one owns – for instance to turn off one's lawn irrigation system, to turn on one's home security camera, or to unlock one's front door – from any terminal.

- The incremental cost of adding and maintaining additional devices should be near zero. For example, although it is acceptable, if not fun, to spend several hundred dollars of time or money installing a $2000 device, the time and money budget for managing sub-$100 devices will be much smaller.

- The systems should be trustworthy [26] by providing strong, understandable security and reliability guarantees.

Sadly, software's ability to automatically manage such ad-hoc collection of devices has not kept pace with hardware advances. Already, management issues are significantly problematic for "early adopters" who own several computers and must keep track of which machine has the most current version of a file, must install software on multiple machines, and must backup all of their data sets for safety. Without a concerted effort to develop an end-to-end software coordination infrastructure, we are in danger of creating a world where people work to service their devices—manually administering them and carrying data from one device to another—when the devices should be serving their users.

The C0PE (Consistent 0-Administrator Personal Environment) project proposes to develop services to reduce the management burden that arises when multiple devices must cooperate in ad-hoc ways. The C0PE project examines two key problems.

1. How to construct a simple security model that can be used by millions of non-expert users to create trustworthy systems each consisting of dozens of devices.

2. How to efficiently export a consistent view of an underlying global data system to collections of devices when the devices have limited or intermittent network connectivity. In particular,

we study ways to resolve the dilemma between a tight coupling of devices' soft state to the underlying global data system to improve consistency and a looser coupling to improve performance.

2 Related Work

C0PE leverages the efforts of a number of current projects that address different aspects of the problem.

C0PE's security architecture draws from IPSec [12], secure DNS [6, 5], and the Simple Public Key Infrastructure [23], but tries to simplify and optimize these protocols for use in a personal environment. Forthermore, Java Jini [11] and Microsoft's Universal Plug-n-Play (UPnP) provide directory services for discovering resources (e.g., available hardware or software interfaces) in ad-hoc collections of devices.

Coda [14], Ficus [22], and Bayou [29] have developed core technologies for disconnected access to data on which we build.

In addition, a number of current projects (Portolano [2], Oxygen [4], wearable computing [28], and Itsy [30], to name a few) are examining different architectural approaches for personal information systems.

3 Security Architecture

A user with dozens of devices must set up an environment where his devices can communicate with each other freely, but where communication with the outside world is more restricted. For example, suppose a user has a digital camera with a short-range wireless network. When the user walks near his portable computer, he would like the data to be downloaded off the digital camera to his portable machine and, when the portable machine docks with a high-performance network, to permanent storage. However, when the user walks near someone else's portable, that machine should not be able to grab the user's pictures from the camera.

Currently, setting up such security arrangements is a complex matter that involves system administration experts and that increases in complexity with the number of devices being considered. To address these issues, we explore two ideas, a "personal virtual intranet" and a "personal virtual firewall". A personal intranet is a simple abstraction that provides unsophisticated users a way to reason about security: devices placed on an intranet are visible to one another and generally trust one another while other devices are less trusted and must be accessed through the personal virtual firewall. The personal firewall captures the same notion that a corporate firewall does—it divides the world into "trusted" and "untrusted" regions and it provides a single point to regulate policy between them. More sophisticated security models might be contemplated for this environment, but we believe that simplicity is essential for making the system usable by non-computer-scientists.

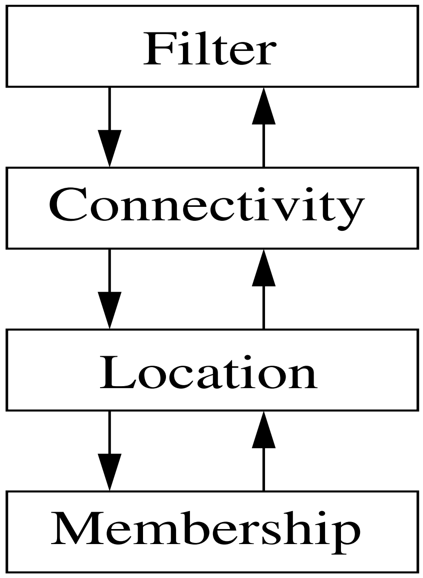

Figure 1: C0PE security architecture.

Also note that while we build the simple abstractions of a intranet and a firewall, the underlying implementation is much different from their traditional counterparts. Instead of securing the network by controlling access to the physical wires at a single connection to the outside world, we must use encryption and policies distributed across machines to achieve the same effect.

The C0PE security architecture consists of four layers as shown in Figure 1.

The Membership layer provides the public key infrastructure for a personal environment. To add a new device, a user first connects to an existing device (either by making a physical contact or via an encrypted network connection to an existing device), and then authorizes the addition of the new device by accepting the request to join at the existing device. The underlying system then signs a certificate authorizing the new device to join the user's C0PE environment [32, 1].

Although public key infrastructures are, in general, difficult to engineer, the C0PE environment simplifies matters considerably because our goal is to assemble a relatively small collection of devices under a single authority.

One key challenge in constructing the Membership layer is dealing with man in the middle attacks. When a new device joins the system by connecting to a user's well-known "C0PE certificate authority" (C0PE-CA), an adversary could represent itself as the COPE-CA to the new device and represent itself as the new device to the C0PE-CA. Although a sophsticated user may be careful to manually examine the certificates presented to each device [23], we suspect that most users will accept most certificates presented to them without examining the content. One approach we are exploring is to restrict our device registration protocols to require (a) direct physical connectivity between the new device and the existing device (e.g., via line of sight infrared or body-area network [33]) or (b) proof of physical contact between the authorizing user and the new device (e.g., a challenge-response protocol in which the existing device prompts the user to enter a typed, written, spoken, or "tapped" [31] phrase into the new device).

The Location layer's task is to allow a user to contact any of her devices or services using a high-level name. This functionality is similar to that provided by the Doman Name System (DNS) [19]: given a name, return the physical address (e.g., IP address) associated with that name. In the C0PE environment, this problem is made more challenging by device heterogeneity, device mobility, dynamic IP assignment (DHCP), and network address translation (NAT). At the same time, the because the system is designed to share each set of mappings among a collection of a few dozen devices rather than across the entire Internet, we can make certain simplifying assumptions.

The C0PE Location layer optimizes name translation for a personal environment in two ways. First, rather than use on-demand hierarchical caching as in DNS, the nodes in a C0PE use eager peer-to-peer replication of name to address mappings. This optimization is made possible by the relatively small scale of each namespace. Second, rather than providing cache coherence via time-to-live values for entries, C0PE treats name translations strictly as *hints*. Because all connections between a user's devices are authenticated, an attempt to connect using a stale mapping will be rejected by the Connectivity layer (described below). In a C0PE environment with dynamically chaninging addresses and intermittent connectivity, this approach allows the system to resolve the dilemma between short time to live values (which are appropriate for dynamic environments but not those with intermittent connectivity) and long time to live values (which are appropriate when connectivity is intermittent but not when mappings are dynamic.)

The Connectivity layer manages connections between devices in the same C0PE. The Connectivity layer uses the routing hints provided by the Location layer to direct packets, and it uses the membership certificates provided by the Membership layer to authenticate connections. The Connecticity layer also consults the Policy layer to accept only connections that (a) originate from within a user's C0PE or (b) are directed to services that have been explicitly designated as publically-available. This approach follows the principles of *complete mediation* and *failsafe defaults* [24]: all incoming requests are inspected and by default all services are restricted to "internal use."

We have developed a Linux implementation of the C0PE Connectivity layer based on IPSec [12]. A key feature of this layer is that by leveraging the Location layer, it allows two nodes, each with dynamically-assigned IP addresses, to securely connect to one another.

The Policy layer determines which devices can access which services. Implementation of this layer is aided by our assumption of a personal-scale environment across which data is replicated. Our initial policy takes a minimalist approach of providing a firewall abstraction. Firewalls are limited in that they provide only two classes of connection – trusted and untrusted. As noted above, more sophisticated approaches have been proposed, but simple firewalls remain one of the most popular security architectures in practice. We believe that a security system intended for use by non-computer-scientists should maximize simplicity. The questions that we seek to address include: (1) does the firewall approach give up too much flexibility in pursuit of simplicity? and (2) are firewalls simple enough for mass-market users to understand and correctly use?

4 Consistent storage abstractions

Our objective in designing the storage architecture for a user's information's space is to approximate as closely as possible the abstraction of a storage service through which all data is acccessible from all devices. The following assumptions impact our design of this architecture:

- Although devices are likely to have large amounts of local soft state (e.g., caches and write buffers), we believe that hard state storage ought to be provided by globally-accessible services running at carefully engineered data centers. This approach reduces the management cost of devices, reduces the risk of losing data if devices are lost or damaged, and provides a cornerstone for providing a consistent view of all parts of a user's data environment.

- Most devices will have network connectivity, but because of power constraints, the connectivity of many devices will be short range or slow or both.

To provide the illusion of universal, high-speed connectivity to a global consistent data store, we are focussing on several issues:

1. *Client cooperation to improve disconnected operation.* We examine algorithms for allowing several devices that can communicate with one another but that are disconnected [14] or weakly connected [20] to the global information system to coordinate their activities to improve the illusion of robust, consistent, high-performance storage. In particular, we build on our cooperative caching work [3] to develop ways for collections of nearby clients to determine near-optimal strategies for data placement [15]. These strategies allow clients to supply efficiently data to one another and thereby improve the effectiveness of caching, prefetching, and hoarding [14, 16].

   Note that in this environment, the effectiveness of these algorithms is measured not only by the hit rates achieved but also by the power or money consumed sending data across wireless links. Additionally, we extend our efforts to allow clients to safely supply dirty data directly to one another without first writing it to permanent storage [21]. We hypothesize that this strategy for propagating writes within a cluster of machines can improve both consistency (devices see more up-to-date data) and fault tolerance (the loss of a device is less likely to cause data to be lost) [18].

2. *Update propagation topology.* In peer-to-peer replication systems (e.g., Reconcile [9], Ficus [22], Bayou [29]) eventually all nodes need to see all writes. A goal in such systems is to distribute writes cheaply and quickly. For example, consider a user with multiple machines at home and at the office, with the two clusters connected via modem. Within each cluster, updates should happen quickly – the user should be able to walk from room to room and not detect that the machines in the cluster are storing different data. At the same time, the system should ensure that updates traverse the modem link exactly once. Similarly, a mobile user's device may need to decide whether it should transmit updates across an expensive cell-phone network and or it should delay the update propagation in hope of encountering a cheaper network later.

   To determine the best course of action, the system needs to take several steps. First, it needs to determine the network's topology. This is challenging both because the network topology changes over time (because of user mobility and changing network loads) and because multiple devices may need to reach some level of agreement on the topology in order to develop an efficient update propagation plan. Once the topology is determined, the system needs to apply an update propagation algorithm that balances consistency, durability of data (to guard against device failure or disk crash), network bandwidth consumed, and/or dollars spent. To address the latter issue, we are developing access-pattern-aware algorithms that prompltly send updates where they are more likely to be needed, as well as algorithms that exploit knowledge of the semantics of the updated data to select the most cost effective update propagation policy.

3. *Cache consistency micro-protocol.* We develop a low-level programming interface for expressing the precise cache consistency requirements of applications. This approach is inspired by Shen, Arvind, and Rudolph's CRF consistency model for hardware architectures, which allows the compiler to specify consistency requirements and for different run-time systems to implement those requirements differently [27]. By allowing applications to specify required semantics rather than programming to the implementation details of a particular system, CRF improves code portability and also allows more opportunities for performance optimization. Both concerns are important for ad-hoc distributed systems. For example, in different file systems today, "close()" can be a performance hint that an application is done accessing data, it can be a consistency command to make data visible to other processes [10], it can be an order to write a single file to disk synchronously [25], or it can be an order to write synchronously a set of files to disk [8]. Separating these notions allows applications to get the guarantees they need without paying for the ones they don't. To make CRF-style consistency useful in this environment, we must (1) extend CRF to include the notions of persistent storage and fault tolerance that are not present in the hardware architecture domain, (2) construct a library implementing this microprotocol using several distributed file systems and distributed object systems, and (3) express higher-level semantic requirements such as those provided by optimistic concurrency [17], Bayou [29], location consistency [7], and Globe [13] in the microprotocol and measure the performance overhead of the added abstraction layer.

5 Conclusions

In the very near future, each individual will own dozens of heterogenous devices and will have to face the non-trivial task of managing them. The C0PE project aims at building an infrastructure that reduces the onus of system administration (i) by allowing these devices to be configured automatically, and (ii) by providing a consistent and unified view of the user data from all devices. Both problems are well known to be hard to solve in their full generality: our approach to address them is to develop techniques that favor simplicity over functionality and take advantage of the unique characteristics of a personal environment, such as the relative small scale of its name space. We are in the process of developing a prototype of the C0PE infrastructure. Our current focus is on designing and implementing the C0PE's security architecture.

References

[1] E. Belani, A. Vahdat, T. Anderson, and M. Dahlin. The CRISIS Wide Area Security Architecture. In *Proceedings of the Seventh USENIX Security Symposium*, January 1998.

[2] G. Borriello. Portolano: Charting the New Territory of Invisible Computing. Technical report, University of Washington Computer Science Department, 1999. http://www.cs.washington.edu/ research/ portolano/ proposal/.

[3] M. Dahlin, R. Wang, T. Anderson, and D. Patterson. Cooperative Caching: Using Remote Client Memory to Improve File System Performance. In *Proceedings of the First Sympo-*

sium on Operating Systems Design and Implementation, pages 267–280, November 1994.

[4] M. Dertouzos. Oxygen. http://www.lcs.mit.edu/news/releases/oxygen040799.

[5] D. Eastlake. Secure Domain Name System Dynamic Update. Technical Report RFC-2137, Internet Engineering Task Force, Apr 1997.

[6] D. Eastlake. Domain Name System Security Extensions. Technical Report RFC-2535, Internet Engineering Task Force, Mar 1999.

[7] M. Frigo and V. Luchangco. Computation-Centric Memory Models. In *Proceedings of the Tenth Annual ACM Symposium on Parallel Algorithms and Architectures*, June 1998.

[8] G. Ganger and Y. Patt. Metadata Update Performance in File Systems. In *Proceedings of the First Symposium on Operating Systems Design and Implementation*, November 1994.

[9] J. Howard. Reconcile User's Guide. Technical Report MERL TR99-14, Mitsubishi Electric Research Laboratory, March 1999.

[10] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West. Scale and Performance in a Distributed File System. *ACM Transactions on Computer Systems*, 6(1):51–81, February 1988.

[11] Jini Technology Architectural Overview. http:// www.sun.com/ jini/ whitepapers/ architecture.html, January 1999.

[12] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. Technical Report RFC-2401, Internet Engineering Task Force, Nov 1998.

[13] A. Kermarrec, I. Kuz, M. van Steen, and A. Tannenbaum. A Framework for Consistent, Replicated Web Objects. In *Proceedings of the Eighteenth International Conference on Distributed Computing Systems*, May 1998.

[14] J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1):3–25, February 1992.

[15] M. Korupolu and M. Dahlin. Coordinated Placement and Replacement for Large-Scale Distributed Caches. In *Proceedings of the 1999 IEEE Workshop on Internet Applications*, June 1999.

[16] G. Kuenning. *Seer: Predictive File Hoarding for Disconnected Mobile Operation*. PhD thesis, The University of California at Los Angeles, 1997.

[17] H. Kung and J. Robinson. On Optimistic Methods for Concurrency Control. *ACM Transactions on Database Systems*, 6(2):213–226, June 1981.

[18] B. Liskov, S. Ghemawat, R. Gruber, P. Johnson, L. Shrira, and M. Williams. Replication in the Harp File System. In *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, pages 226–238, October 1991.

[19] P. Mockapetris and K. Dunlap. Development of the Domain Name System. *Computer Communications Review*, 18(4):123–133, August 1988.

[20] L. Mummert, M. Ebling, and M. Satyanarayanan. Exploiting Weak Connectivity for Mobile File Access. In *Proceedings of the Fifteenth ACMSymposium on Operating Systems Principles*, pages 143–155, December 1995.

[21] S. Rao, L. Alvisi, and H. Vin. Low-Overhead Protocols for Fault-Tolerant File Sharing. In *Proceedings of the Eighteenth International Conference on Distributed Computing Systems*, pages 452–461, May 1998.

[22] P. Reiher, J. Heidemann, D. Ratner, G. Skinner, and G. Popek. Resolving File Conflicts in the Ficus File System. In *Proceedings of the Summer 1994 USENIX Conference*, 1994.

[23] Ronald Rivest and Butler Lampson. *SDSI - A Simple Distributed Security Infrastructure*, 1996.

[24] J. Saltzer and M. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9), September 1975.

[25] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. Design and Implementation of the Sun Network Filesystem. In *Proceedings of the Summer 1985 USENIX Conference*, pages 119–130, June 1985.

[26] F.B. Schneider. Information Systems Trustworthiness - Interim Report. Technical report, National Research Council/National Academy of Sciences Computer Science and Telecommunications Board Commission on Physical Sciences, Mathematics, and Applications, April 1997.

[27] X. Shen, Arvind, and L. Rudolph. Commit-Reconcile & Fences (CRF): A New Memory Model for Architects and Compiler Writers. In *Proceedings of the Twenty-Sixth International Symposium on Computer Architecture*, May 1999.

[28] D. Siewiorek and A. Smailagic. Software Development for the Navigator Wearable Computers. In *Proceedings of the Portable by Design Conference*, February 1994.

[29] D. Terry, M. Theimer, K. Petersen, A. Demers, M. Spreitzer, and C. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proceedings of the Fifteenth ACMSymposium on Operating Systems Principles*, pages 172–183, December 1995.

[30] M. Viredaz. The Itsy Pocket Computer Version 1.5: User's Manual. Technical Report TN-54, Compaq WRL, July 1998.

[31] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. The parctab ubiquitous computing experiment. Technical report, Xerox Palo Alto Research Center, 1995.

[32] E. Wobber, M. Abadi, M. Burrows, and B. Lampson. Authentication in the Taos Operating System. *ACM Transactions on Computer Systems*, 12(1):3–32, February 1994.

[33] T. Zimmerman. Personal Area Networks: Near-field intrabody communication. *IBM Systems Journal*, 35(3&4), 1996.