Supporting Research Collaboration through Bi-Level File Synchronization

Catherine C. Marshall, Ted Wobber, Venugopalan Ramasubramanian, Douglas B. Terry

Microsoft Research, Silicon Valley
{cathymar, wobber, rama, terry}@microsoft.com

ABSTRACT
In this paper, we describe the design and use of Cimetric, a file synchronization application that supports scholarly collaboration. The system design incorporates results of earlier studies that suggest replicating content on a user's personal devices may have different characteristics than replicating content to share it with collaborators. To realize this distinction, Cimetric performs bi-level synchronization: it synchronizes local copies of a versioned repository among collaborators' computers, while it separately synchronizes private working files between each user's personal devices. Through a year's worth of in-house use of Cimetric in a variety of configurations, we were able to investigate key file synchronization issues, including the role of cloud storage given the ability to sync between peers; the strengths and weaknesses of a bi-level design; and which aspects of the synchronization process to reveal to users.

Categories and Subject Descriptors
H5.3. [Information Systems]: Information interfaces and presentation---Group and Organization Interfaces

General Terms
Human Factors, Design

Keywords
File synchronization, cloud storage, scholarly collaboration.

INTRODUCTION
In recent years, file synchronization services have been identified as key to working across multiple devices [8, 14, 26, 27], to collaborating with colleagues [9], and to keeping files safe by replicating content in different locations [16]. Consumer-oriented products such as Groove [10], Dropbox [9], Google Drive [12], and Windows SkyDrive [35] acknowledge the varying roles of file synchronization in heterogeneous computing environments in which people access (and potentially edit) content on the device at hand, taking advantage of the available level of network connectivity.

Yet the very people who might benefit the most from file synchronization technologies have been slow to adopt them. According to a study by Dearman and Pierce [8]:

> *"Our findings suggest that people do not trust automatic file synchronization, even though they employ automatic synchronization for other types of information: music, email messages, contact information, calendar data, and task lists."*

We were interested in exploring the adoption and use of file synchronization technology by observing it in action. To do this, we developed and fielded an application, Cimetric, that would appeal to a local community by addressing a common activity, research collaboration and paper-writing.

In particular, we sought to address an aspect of file synchronization that has been identified in previous studies, the distinction between sharing files with oneself and sharing files with one's collaborators [11, 17, 24]. While the same general infrastructure can support both forms of sharing, the rhythm of synchronization in each case is different: authors may want to replicate incomplete drafts among their own devices while they are working on them, and share them when the writing is in a more intelligible state, ready for their collaborators' attention. They may also work with a different set of files than they share; collaborators may rely on their own datasets, analysis tools, and editors to address distinct parts of a complex task [17].

We used a topology-independent replication platform [25] as the basis for implementing Cimetric. Because the platform allows files to be synced between peers, we were able to explore the role of cloud storage in a collaboration in which some coauthors are co-located and others are distant, and some work is almost synchronous, while other work is spread out over time. Would co-located collaborators be able to take advantage of the efficiencies of peer-to-peer synchronization? What kind of feedback would be necessary for distant collaborators to know whether their local files were up-to-date and who was currently working on them? Would a design that distinguishes between syncing personal files on one's own computers and sharing files with colleagues better support collaboration or would the added complexity of a bi-level sync be a burden? Observations of Cimetric in use over time and in support of real work helped us answer these questions.

This paper begins by discussing related work, including studies of file sharing, products that are currently used by people engaged in various sorts of scholarly collaboration, and previous work on collaborative writing. We then describe Cimetric, the application we developed to reflect our understanding of scholarly collaboration and local needs. After these background elements have been laid out, we describe our observations of Cimetric in use over the course of a year for collaborative writing and other related file sharing activities. Finally, we summarize what we have learned and evaluate our efforts against our original aims.

GROUP'12, October 27–31, 2012, Sanibel Island, Florida, USA.

RELATED WORK

There are three types of related work to consider: Studies of file sharing; research prototypes and products that support file synchronization; and studies of collaborative writing, particularly those in performed academic environments. The first two types of related work will be the most salient in identifying findings for our work; the third type of related work feeds into our discussion of Cimetric's design.

Studies of file sharing

Although our work can take advantage of the lessons learned by studies of general file sharing, for example, that awareness of the activities of people who are not actively collaborating may be useful to the group [21], or that there are different aspects of activities collaborators need to be aware of [33], we are the most focused on sharing that involves the co-creation of content in versioned systems. To this end, Fitzpatrick et al. describe different types of CVS events (for software developers and beyond) which need to be brought to their group's attention [11] and Yamauchi et al. discuss the role of CVS repositories in successful collaborations [36]. In general, however, we are much more narrowly focused on synchronization events.

Voida et al. [30] discuss general practices related to sharing files, identifying problems such as choice of sharing service and naming recipients; most salient to our work is their discussion of breakdowns, including several problems we anticipated such as the continued need for out-of-band notification to highlight new or changed content. This is a general problem for file syncing, and, as we discuss, we experimented with a variety of ways of surfacing changes, and documenting file location and provenance.

Although we are aware of the eventual need for introducing security mechanisms [33], we are focused most closely on the design and use of Cimetric's synchronization machinery and the feedback it offers users about its status.

File syncing systems and products

General synchronization research has been the province of the systems community (e.g, [4, 25, 26]); until recently, the results of this work have not played a role visible to users. Of particular interest is Perspective, a decentralized storage system for home use and fielded in homes [26]; one important difference, however, is that Perspective is not designed to support evolving content.

Systems work in the HCI/CSCW community has long been focused on shared online repositories (e.g. [5, 21]); needless to say, by now there are many more research systems and products. Instead of taking an approach that relies on a centralized server- or cloud-based remote repository, we are focusing on a synced local store to support sharing.

Because we are interested in real use (albeit in a local setting so it is easily observed and supported), we compare our approach with three types of products our users might consider in lieu of Cimetric: Dropbox, Google Docs, and popular distributed version control systems such as Git (and online GitHub repositories).

Dropbox is a widely adopted application that syncs local files among devices using the cloud as an intermediary that maintains file versions, which can be accessed through a Web browser. While Cimetric has some of the same functionality as Dropbox, we are interested in seeing the effect of an optional cloud store, as well as investigating an explicit distinction between working files and shared files. Finally, while Dropbox hides much of its sync mechanism (users may not be aware of how Dropbox works, conceiving of it as a literal dropbox or a cloud-only store [18]), Cimetric allows users to inspect and control many aspects of synchronization, so they can address, for example, bandwidth limitations and differences in work style.

Google Docs supports on-line synchronous editing of cloud-resident documents. Although it is a popular tool for the co-creation of content, unlike Cimetric or Dropbox, it requires always-on connected operation to support content changes, as well as the adoption of specific editors.[1]

Cimetric shares some functional aspects of source/revision control systems (e.g. CVS [7], Git [1], Mercurial [29], and Subversion [2]), including provenance-tracking, offline working sets, and asynchronous updates to shared state, although Cimetric tracks provenance on a per-file basis only and does not group edits to multiple files as a single changed version. More to the point, because it was designed for general collaboration, Cimetric omits features of revision control systems that are aimed at software development (e.g. branching, automatic merging, and exclusive locking); our prior study suggested that the complexity of a revision control system was likely to require more administrative overhead and intellectual effort than many ordinary users in our target environment would tolerate [17].[2]

Studies of collaborative writing

Although our work is not aimed at extending the scope of previous collaborative writing research [1, 23, 15, 19, 32], we rely on this research to inform our understanding of some salient local work practices and perspectives (as documented in [17]). As these studies have shown, collaborative writing is largely asynchronous (although it may become more synchronous as deadlines approach), crucially involves email for draft-passing, and is tied to the authors' normal content production tools.

CIMETRIC SYSTEM DESCRIPTION

As a result of our understanding of the intended use situation and our need to more closely observe the problems that arise from file syncing [3, 27], we developed an application called Cimetric. Cimetric is a Windows application that manages documents and other files associated with ad hoc collaborations. It serves multiple purposes: collaboration (sharing data with others); roaming (sharing data with oneself on different devices); and backup (copying data to secondary storage).

We designed Cimetric to have the following key attributes:

- Decentralized synchronization that does not rely on a single authority (such as a central server), and that can handle the demands of offline operation;
- Bi-level synchronization that distinguishes between sharing work in progress with oneself across devices, and sharing versions of these files with one's collaborators;

[1] Its successor, Google Drive [12], has some support for offline operation; however, unlike Cimetric, access to shared files that the user does not own requires network connectivity.

[2] In reported past experiences, version control systems were adopted by some members of a collaborative writing group, and not by others (because of their apparent overhead), thus thwarting their original purpose.

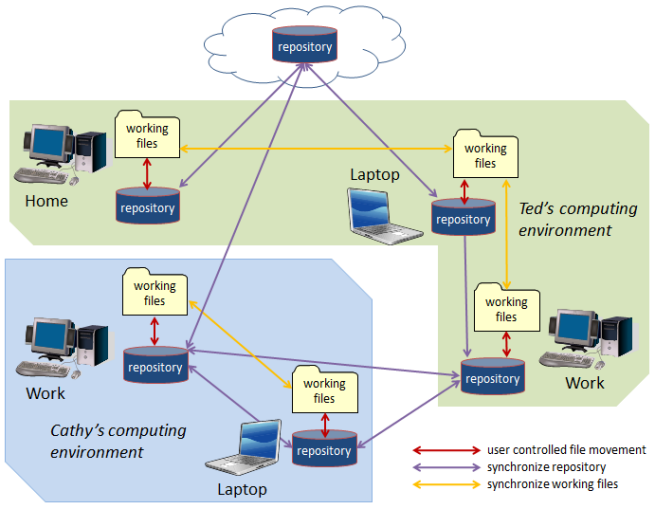

Figure 1. An example Cimetric configuration.

- User control of change integration (in other words, preventing the synchronization process from accidentally destroying local changes by overwriting them);
- Sufficient feedback to allow users to understand the system state (e.g., whether synchronization has completed when one is about to go offline);
- Lightweight mechanisms to show users what their collaborators are doing (e.g. who last wrote a file, or who is working on it now); and
- The ability to use cloud storage without relying on it.

The remainder of the section describes Cimetric's abstractions, architecture, and the feedback it offers users.

Abstractions

Cimetric is based on three central abstractions: *collaborations*, *worksets*, and *repositories*. From a user's perspective, each collaboration has two parts: (1) a *workset*, which contains a user's working files for a particular effort, and (2) a corresponding *repository*, which is the set of files that are shared among the people who are working together. A *workset* is a folder (or a folder hierarchy) in the file system which is managed by the user, and which may be replicated among the user's computers. Files in a workset are not versioned; they look and behave like normal Windows files. A *repository* is a versioned file store that is managed by Cimetric. A user controls the movement of files between workset and repository and vice-versa via a lightweight mechanism that allows users to communicate with each other about who is working on a file (or files); this mechanism also enables users to incorporate their colleagues' changes when they are ready, so their own changes are not overwritten.

From the system's perspective, a *collaboration* may be hosted at multiple client computers; the repository that stores the collaboration's constituent files is replicated in full on each computer. Worksets—i.e. a user's working files—may be replicated as well, so users can continue working seamlessly as they move from, say, a work computer to a laptop, then to a home computer. A user may be involved in multiple collaborations (possibly with different collaborators) using the same instance of the Cimetric application.

A repository instance may be hosted in the Azure cloud [6] if desired, but the cloud is not required by the application. For example, collaborators may decide to create an Azure repository instance because one person works outside the firewall. If several collaborators are on the same subnet, and are working nearly synchronously because a deadline is approaching, they may prefer to use Cimetric's peer-to-peer synchronization because it is significantly faster and lower overhead than syncing via the cloud.

Architecture

Figure 1 shows the Cimetric architecture. In this example, Ted and Cathy are working together on a paper. Ted's computing environment includes a home computer, a laptop he works on while he commutes on the train, and a computer that he uses at work. Cathy is only using two computers to work on the paper, her laptop, and a work desktop. Ted's working files (his Cimetric workset) are replicated on each of his three computers, but as Figure 1 shows, his laptop usually syncs with his home computer when he brings the laptop home, and with his work desktop when he brings the laptop back to work. Likewise, Cathy's working files are replicated on each of the computers she's using to write the paper. Both Ted and Cathy may have files replicated on their own computers that the other doesn't see; practically speaking, these files may include source content related to a particular activity each is working on alone, say creating the figures, or temporary files generated in the course of writing—for example, intermediate versions, that aren't suitable for sharing, or PDFs of references one author is reading for the purpose of filling in citations.

Each computer involved in the collaboration also has a complete local copy of the repository. Like worksets, repositories are synchronized with one another opportunistically. In Figure 1, Cathy's repositories sync with one another, and with the repository instance on Ted's work computer. The repository instance on Ted's laptop syncs with his work computer too. In this scenario, Ted and Cathy have discovered that it would be convenient to have a repository instance in the cloud because they tend to work outside of the firewall fairly frequently. Not all repository instances must sync with the one hosted in the cloud though—only Ted's home computer and laptop and Cathy's work computer sync with the cloud.

We have discussed how worksets sync with selected partners, and how repositories similarly sync with one another. How do files move between the two local stores? Users control the movement of files as they move between a workset and the corresponding local instance of the repository by using an explicit, user-initiated mechanism. Each time a user selects files from the repository and moves them to his or her workset, the mechanism overwrites the existing versions of the files; the system asks the user's permission if a newer version is being replaced by an older one. When a user moves files from a workset to a repository, new versions of the files are created in the repository instance. Older file versions can be moved from repository to workset at a user's request; they also can be inspected in place if, say, a user wants to recover specific text. The portion of the repository browser that provides access to older file versions is not prominent, however. It relies on user discovery, because we recognize that it is relatively uncommon for a user to return to an earlier file version.

Versions are visible (on demand) to users for three reasons. First, they protect a user against accidental loss. Because we are urging researchers to use the system for real, time-critical efforts, we are being conservative about potential content loss, regardless of its source (user error, system malfunction, or design infelicities).

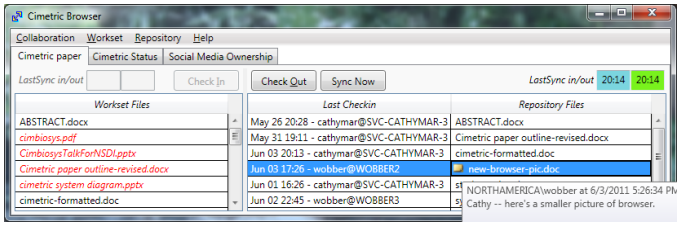

Figure 2. Cimetric Browser

Second, our system allows users to modify shared files when disconnected and, in general, to make updates asynchronously. Explicit versioning makes it possible to coordinate such modifications and to detect conflicts. Finally, system-supported versions enable us to probe the utility of specific aspects of the version abstraction from a user's perspective: will users ever retrieve content from old versions? Will they replace newer versions with older ones? Will they consult them to resolve conflicts? Will they freely overwrite their collaborators' efforts, knowing that no content is destroyed?

As Figure 1 shows, files may be checked out of or into any local repository instance, and they need not be checked into the same local repository they were checked out from. To accomplish this, workset files must carry information about their provenance at the repository of origin. At check-in time, the provenance of the workset file is compared with that of the version (or versions) in the target repository. If the former is not a superset of the latter, a conflict exists. This same mechanism allows for the detection of versions submitted simultaneously to different repositories. Since Cimetric allows simultaneous updates, the best that we can do is to show the conflicting versions to users and let them resolve the conflict after the fact.

Cimetric uses the Cimbiosys topology-independent synchronization protocol via its replication library [25]. Practically, we could have used other popular sync technologies that offer an SDK to support application development, but as we explained earlier, we were interested in exploring certain features of Cimbiosys (for example, decentralized peer-to-peer synchronization) through use. Cimbiosys uses Windows Communication Foundation (WCF), and the default mode of communications uses TCP connections and the TCP protocol. Cimbiosys guarantees *eventual consistency*. In other words, over time the replicas will converge and each will store the latest file versions; its design assumes that network connectivity will be intermittent, and that users may wish to work without cloud storage (for reasons such as privacy or performance).

It is often difficult for users to set up peer-to-peer network connections. Because we didn't want to burden users with the need to pass around complex network addresses to join collaborations, a broadcast protocol enables Cimetric instances to discover each other when they are on the same network segment; user-assigned names allow people to identify the appropriate collaboration. Thus, instances on the same network segment can establish connections with one another easily. Alternatively, there is a user interface for inserting the URL of a designated collaborator by hand (in this case, we might expect collaborators to email each other the URLs of their own instances). Finally, the cloud can be used as a central point for establishing a potential sync connection between local repository instances.

User interface

Most of the time, Cimetric does not need to be a visible part of collaborators' work; as is the case with Dropbox and other sync infrastructures, people interact with a synced folder in the file system the way they would normally, editing the files with their usual editors, and managing the files through the folder hierarchy. Cimetric can sync a user's working files in the background, keeping them up to date with a sync partner if the user works on multiple computers capable of syncing with one another. That way, it is easy for a user to switch among devices.

As an aspect of our investigation, certain aspects of Cimetric's operation are revealed through a browser, shown in Figure 2. Each browser tab corresponds to a collaboration (which may involve different people and computers). Workset files are listed on the left; repository files are listed on the right. The hierarchical structure of the workset is represented in terms of paths; in the collaboration shown in Figure 2, no subdirectories were used. Repository files show when they were last checked in (and by whom, from which computer); comments and advisory locks are shown if they exist. The advisory locks do not prevent other users from using the files—they are simply a visual indication that someone else may be modifying the same file; we expect this type of conflict to be resolved socially.

The workset file list uses visual conventions to indicate whether the user has changed the file since the last checkout, or whether there is a newer version in the repository. If both conditions are true (the user has changed the file and there is a newer version in the repository), then a user-generated conflict exists. We leave it to the user to resolve such conflicts; automatic resolution is apt to result in a merge that does not take user intent into account. Repository conflicts are likewise indicated, and are left to the collaborators to resolve. For example, if two people check in new versions of the same file, a repository conflict will result, and will require human attention.

The browser also gives the user access to the sync process. A user can initiate a manual sync and can get answers to questions such as "When did my repository and workset last sync? Where is the workset folder stored? Who is working on what file, and what are they doing?" Other functionality, hidden more deeply in the UI, allows the replication-savvy user to control parameters such as the sync interval (how often the system attempts to sync with its peers), or to turn automatic sync off and work with user-initiated sync. Logs provide an additional means of inspecting what has happened during the sync process.

As is true with some popular file synchronization services, a history mechanism allows users to inspect a file's history and retrieve older versions of it; like most version control systems (and unlike most file sync services), the versions are created explicitly when they are moved into the repository, with the idea that the versions that are shared with collaborators are more meaningful than the versions created by automatic workset syncs. Older versions may be examined or users may move them into their worksets (in which case, users are consulted to make sure that their intention was to overwrite a more recent version of the file).

Cimetric's user interface is designed to meet the expectations of its immediate audience, people who are generally familiar with synchronization concepts. If the system were to be used by a broader audience, some aspects of its functionality might be more readily accessible through visualizations (for example, of file movement during sync); others may end up being hidden from the casual user. Conversations with our user community, as well as

the results of a broader study [18], indicate that synchronization has been rendered overly opaque as it stands. Changes to the user interface have been iterative, and have relied on continued feedback from the user community.

OBSERVING CIMETRIC IN USE

For Cimetric to be adopted and used locally, it needed to address a real problem and be sufficiently reliable for people to use it in the face of deadlines. Potential users needed to be assured that their data was versioned and safe. Crash recovery needed to be simple, and involve only a restart of the application. Furthermore, the application needed to be easy to use with existing material, and easy to opt out of if it didn't satisfy a group's needs.

In this section, we describe how we fielded Cimetric and what we learned from doing so. We recruited real users in our own organization; we wanted to be able to support these users and observe their collaborations closely. We knew that if Cimetric proved to be useful, the researchers who used it would pull in additional collaborators and there would be more adoption as time went on.

To recruit users, we gave talks and demos to describe Cimetric and what it might be used for; we also talked to people who were in situations that might benefit from the application (e.g. writing papers and sharing data files). To discover how the system was being used, we relied on a multi-dimensional approach: (1) we supported users (in person and via email), with an eye toward finding out what they were doing with the system; (2) we engaged in iterative design, creating frequent releases of the application with new user-driven features and bug fixes so that users felt their needs were being met; (3) users sent us feedback, both to influence system design, and to be good citizens; and (4) we interviewed users during and after they used Cimetric, using their own repositories to elicit responses. Thus our data consists of notes taken during observed use; recorded interviews; the file repositories themselves (examined with the users' permission); system logs; and accumulated email correspondence.

The observation period has lasted about a year, and has involved 9 distinct collaborative activities (summarized in Table 1). One of the collaborations, UC9, was active until recently, and two of the others, UC3 and UC5, still see intermittent activity. In all, there were 12 different users involved in the 9 collaborations; in five of the collaborations, a member of a Cimetric-based collaboration used the application for a second or third project.

Use characteristics

So far, adoption has been dominated by dyads, pairs of collaborators sharing files, although four-person and three-person collaborations used the system too. One singleton also used Cimetric to replicate his own files among multiple devices (much as he used Live Mesh, a predecessor to Windows Skydrive [35], earlier). Although we described the application to prospective users as collaborative, we felt that single-person adoption might be a viable way to encourage collaborative use when this user began new collaborative projects.

Table 1 summarizes the use cases we have been tracking in the field. Although we requested that internal users let us know when they installed the system, people in other organizations inside our company could also install it and use it without our intervention (and the existence of several mystery repositories leads us to suspect they did). Table 1 also indicates whether the collaboration used a cloud replica, and whether an external collaborator was involved.

ID	Description	# of users	External collaborator	Cloud replica?
UC1	Sharing project-related files	4	No	No
UC2	Writing a paper	2	No	No
UC3	Sharing project-related files	2	Yes	Yes
UC4	Writing a paper	2	Yes	Yes
UC5	Writing a paper	4	No	Yes
UC6	Writing a paper	2	No	Yes
UC7	Developing algorithms	1	No	No
UC8	Writing a paper	3	Yes	Yes
UC9	Writing a paper	2	Yes	Yes

Table 1. Summary of observed Cimetric use

The uses were by-and-large successful: in 7/9 cases, the desired collaborative artifacts were created, and people did not lose their data when the system occasionally crashed (with one exceptional situation we will describe later in this section). In one case, UC3, the prospective Cimetric users switched to email, and in another case, UC9, the users switched to Dropbox when they were revising their paper; both changes were linked to cloud malfunctions.

The range of uses we observed enabled us to address key concerns about our strategy for supporting file sync and sharing, including:

- The role of the cloud;
- The efficacy of a bi-level design; and
- Which aspects of synchronization to reveal.

We have continued to encourage people to use the system now that it is stable and has been developed to the point that it is useful to different kinds of research collaborations. In the future, we would like to observe larger collaborations.

Inherent risks of synchronization

UC7 used Cimetric for four months in a configuration we did not intend; he used the system by himself to replicate files between his office computers so he would not need to store intellectual property on an outside provider's cloud (for example, Dropbox, uses Amazon's S3 cloud service). Before Cimetric, UC7 had also experimented with Microsoft's Live Mesh to sync files between his own computers. Herein lies a cautionary tale.

Dearman and Pierce warn us:

> *"We believe that the lack of trust in automatic file synchronization is due in part to the higher cost of failure. If a user loses an email or a calendar entry, the consequences are relatively minor, whereas losing a file that contains hours of work is much more traumatic."* [6]

UC7's synced files included a subdirectory that contained a critical presentation related to his project. This subdirectory was replicated on two out of three of his office computers. Earlier he had used Live Mesh to replicate the subdirectory on all three of his office computers, as well as in the cloud. As he worked on the presentation in a last minute push before a conference, the file mysteriously vanished. He was justifiably upset: what had happened?

Figure 3 shows the drawing he created on his whiteboard to explain his model of how the sync applications interacted with one another. The "x" represents the folder that contained his lost presentation; "SkyDrive" is the Live Mesh cloud store, and 1, 2, and 3 are his computers. The left (red) box shows the folders on 1, 2, and 3 that are synced by Live Mesh, and the right (blue) box shows the folders on 1 and 2 synced by Cimetric. Interaction between the two systems' conflict resolution code apparently caused the file vanish. Naturally, the undesired delete propagated (the way it would in any replicated system). Although UC7 was subsequently able to recover a recent version of the file using a Windows 7 feature, this mishap still served as a visceral reminder of Dearman and Pierce's warning.

Because synchronization is usually fairly silent once it has been set up (i.e. in systems like Dropbox, LiveMesh, and Cimetric, the synced folder looks like a normal folder), we might expect this type of interaction between sync applications to be relatively common. It is hard for users to remember which folders have been synced, and it's easy to imagine installing sync apps on top of each other, especially if the original sync app did not perform as expected. In other words, *if one solution doesn't work, a user will probably try to solve the synchronization problem a different way, possibly without uninstalling the first solution*. Sync applications that are simultaneously applied to the same folder structure may lead to unexpected side-effects or misidentified update conflicts.

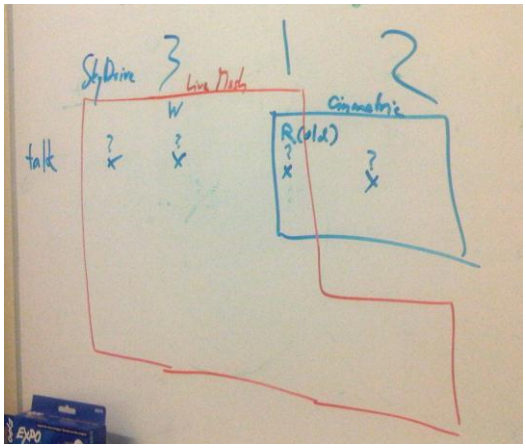

Figure 3. UC7's account of replicated system interactions

The role of the cloud

A significant proportion (6/9) of the collaborations used a *cloud replica*, a copy of the shared repository that was stored in the cloud and synchronized with other repository replicas as a peer. Using the cloud as a peer (rather than as an integral part of the synchronization architecture, as it is in many sync tools) turned out to be an effective way of incorporating the cloud without relying on it. A brief initial period of use revealed that the cloud would be necessary for three reasons:

- Some of the research collaborations involved external colleagues who worked outside the firewall, a situation that made setting up a peer-to-peer network difficult.
- Changing and unanticipated configurations made the cloud an on-demand bridge to connect sync partners that may have been peer-to-peer in the past.
- Power saving software, which is increasingly common, sometimes made the cloud necessary to ensure a path from one PC to another, even during normal working hours (i.e. PCs were often automatically powered down if they were left unattended, even for a brief period of time).

Making the cloud optional, on the other hand, was useful for four countervailing reasons:

- When collaborators were physically proximate (working on the same subnet), peer-to-peer synchronization had better performance; syncs seemed instantaneous to users during periods of intensive semi-synchronous work.
- If the cloud wasn't working (or the connection was slow), it was still possible to sync some users' files. This situation came to pass several times during the longer-running collaborations.
- Collaborations with security concerns did not want to store data on an externally-owned cloud. In this case, the cloud was not externally-owned, but often sync tools use an external—or third party's—cloud services.
- Cloud services may incur additional costs; the cloud may be eliminated from the network when it isn't necessary. In this case, we absorbed the cost of the cloud for the prototype, but we were aware of storage and transaction costs silently accruing, and at a larger scale, these costs could have been noticeable to an organization.

Initially we fielded the system without a cloud replica, in line with the decentralized eventual consistency model assumed by the Cimbiosys platform [25]. Indeed, the system's flexible peer-to-peer topology is a distinguishing feature of the platform, one we were anxious to test. This completely decentralized strategy worked for some collaborations (and for other collaborations *some of the time*), especially for those involving co-located researchers connected to the network during overlapping time periods. In other words, people were not necessarily working on content synchronously, but their computers were connected to the network in such a way that there was an eventual path from one computer in the collaboration to another.

Without the ability to create a cloud replica, adoption was slow. Although the system provided collaborators with certain advantages (the ability to track who was working on what files, for example, and the ability to instantaneously and efficiently synchronize files among personal machines), the advantages weren't so profound that they outweighed researchers' reluctance to adopt a new technology when a deadline was in sight. Furthermore, corporate IT had steadily rolled out new automatic power saving software, so it became increasingly difficult to reach any replicas if a user was trying to sync files when her coworkers were away from their desks (including times during the day when machines powered down because people were in meetings). Without a cloud replica, one's collaborators all needed to be within the corporate firewall at least part of the time, and much collaboration in the lab (and 4/9 of the Cimetric collaborations) involved at least one academic collaborator who was always outside the firewall. The ability to span the firewall turned out to be a significant advantage in enlisting users, and was necessary off and on throughout a collaboration.

It is telling that although we began writing this paper using Cimetric in a wholly peer-to-peer fashion, by the time we were finished, we found it necessary to create a cloud replica of the repository we used to store our work. Normally we all work in the office, but one day close to the deadline, several of us ended up

working from home, and our sync patterns changed abruptly; as with several of the other collaborations, we needed the optional cloud replica midstream. Interview data suggests UC2, one of two collaborations that were successfully completed without a cloud replica, would have created one if the capability were available. Instead, the dyad temporarily shifted to email, and shifted back to Cimetric when they returned to the office.

This pattern of shifting in and out of cloud-based syncing appeared in multiple collaborations as collaborators were added (or dropped out of the collaboration) and as cloud availability changed. UC8 started with a cloud replica; one collaborator was outside the firewall, and the cloud replica was necessary. During the final revision cycles, the cloud was unavailable and the external collaborator was out of the picture. Rapid revision cycles continued via a peer-to-peer connection as the remaining two collaborators worked closely.

Furthermore, it is easy to forget that there may be storage and transaction costs associated with cloud-based repositories; a peer-to-peer solution is, for all intents and purposes, free. Although storage is generally cheap, other demands—e.g., maintaining an open connection to the cloud so repositories can sync at the specified frequency—can make the costs of offering such a service conspicuous.

Evaluating the bi-level design

One of our central research questions concerned the efficacy of the bi-level design: did Cimetric's bi-level synchronization support collaboration in a useful way, or did its complexity overwhelm any potential benefits?

There are several different ways we can reflect on this question: (1) We can examine how people configured the system—did they ever use synced worksets in conjunction with synced repositories in configurations akin to Figure 1? If they didn't, did they adopt workarounds that created a bi-level configuration similar to the one the system supported? (2) Did users perceive any advantages to the system behaviors that stem from this capability, and if not why not? As part of this question, we discuss two related aspects of bi-level design: repository versioning, and controlling the scope of what is shared.

Configurations. We were aware at the outset that people might configure Cimetric in a variety of ways: e.g. one user might not sync worksets (because she writes on a docked laptop that she carries with her), while another might rely on workset synchronization (because she uses a desktop with a big screen in her office, and carries a laptop home with her). We anticipated that at least some of our users would find synced worksets useful. So at first blush, the question seems simple: did anyone (besides UC7, the singleton user we have already discussed) configure Cimetric to take advantage of workset replication?

The answer is, in fact, not simple. In principle, Cimetric users could have used workset replication more often than they did; records show that users often accessed Cimetric repositories from more than one PC. So we asked them why they didn't take advantage of this facility. Three reasons emerged: (1) the lack of a cloud-based workset replica; (2) users' need to reconfigure their computing resources on-the-fly; and (3) the complexity of configuring bi-level sync.

The first reason parallels our initial problems with repository syncing: that is, worksets weren't replicated in the cloud; they relied on peer-to-peer syncing. Thus all of the problems we discussed in the previous subsection were true of worksets as well as repositories.

The second reason was more nuanced: we did not foresee just how often users would need to change configurations on the fly. In other words, although it was easy for new people to join a collaboration (and later, to add a cloud replica to an existing collaboration), it was more difficult for users to change a system configuration mid-collaboration. The design precluded adding a sync relationship between existing worksets or changing the way an existing sync relationship was set up.

For example, one collaborator in UC2 wanted to add another workset/repository pair when he found himself working on his laptop (which he didn't often use) in an unexpected place—a café—outside the firewall. Before a cloud-based repository replica was available (as was the case in the system's early days), this type of improvisation was completely impossible. Even so, if there wasn't already a replica in the cloud, nothing could be done to establish one in this situation short of contacting collaborators to find out if they were in a position to set one up. Finally, there was no provision for the remote worker to access his existing workset.

Thus to take full advantage of bi-level sync—and this speaks to the third point—users had to plan their work and anticipate the computers they would be using and places they would be working. In retrospect, we realize this is a lot to expect. The problem was not so much that Cimetric users did not write on multiple computers, but more that if they did, they did so in an opportunistic way. The ability to synchronize a workset with the cloud would have enabled the system to better handle these unanticipated configurations; that way, if a user found him- or herself working unexpectedly outside the firewall on a different computer, his or her working files would still be accessible (in addition to the files that had already been moved into the repository).

Workarounds. It is possible for users to set up bi-level synchronization themselves. Ironically, one of the collaborators in UC9 was unaware of Cimetric's ability to sync worksets, but he perceived a need to do so. So instead of using Cimetric to sync his workset folder (i.e. his work-in-progress), he used Dropbox to sync this folder and Cimetric to sync the repository he shared with his co-author. Thus he effectively simulated the bi-level design by using the two sync services—Cimetric and Dropbox—in tandem.

Repository versioning. While worksets were normal Windows folders, repositories were versioned; each time a user shared his or her work, a new version of the file would be created. Earlier studies documented that user communities like ours want versions to be maintained on their behalf [15, 17, 23]; popular sync and sharing applications such as Dropbox and Google Docs support for automatic versioning (with the important distinction that in those services, versions are created when files are saved, not when they are explicitly shared). Code development efforts [11] or compliance- and recovery-oriented solutions [20] also make productive use of versions. Would these results generalize to Cimetric?

Again, the answer varied with the collaboration. In some Cimetric repositories (e.g. UC1, UC2, and UC5), the co-authors maintained the same naming conventions they had in the past: collaborators passed drafts back and forth, appending their initials to indicate who had checked in the draft and renaming the file to note the version's role in the overall writing process. In others (e.g. UC9),

the users let Cimetric's file versioning do a larger proportion of the provenance maintenance work for them.

For example, both authors in UC2 were asked about apparently branching versions that used naming conventions (for example, one collaborator's file, *intro.tex*, would be revised by a second collaborator, and saved as *intro-svr.tex*). One of the UC2 authors said, "*Oftentimes when I have to revise a section, I would revise it in a separate file so that if for some reason he wants to go back to the old stuff, he can do it*." The other author in this collaboration adopted a similarly conservative approach and retained all changed text as comments in the shared file. "*You don't take away text*," he explained. In other words, appending one's initials is a way of acknowledging the contingent nature of the changes a co-author introduces. On the other hand, UC9's collaborators, who worked together over a much longer period to create a monograph, simply overwrote one another's files, and relied on the system's versioning mechanism to keep things sorted out. In practice, none of the collaborators ever returned to old versions of their files, but the ability to do so (at least in theory) was comforting.

Controlled sharing. Bi-level design also enabled users to control the scope of what is shared. In other words, users could sync work that they had no intention of sharing (e.g., intermediate files used to produce figures, temporary files created in the process of running LaTeX, and data used to produce the results that appeared in a publication). This aspect of the bi-level design was successful; many users created files they did not share.

Revealing synchronization processes

As Cimetric development progressed, we experimented with different models of which aspects of synchronization to reveal, and which aspects could remain hidden. Certainly revealing too much was as confusing and as unhelpful as revealing too little. We kept the design of this feedback literal, with the idea that an adept designer could generalize from what we've learned, and potentially create better visualizations of portions of the sync process.

From interviews and observing use, we found that it was useful for the tool to reveal three types of information: information about file provenance; the status and progress of the sync itself; and an overview of what has changed.

Provenance. Where did a file come from? When was it last synced? Users found this information to be useful to track their own activities as well as their collaborators' work. Figure 4 shows a portion of a Cimetric browser that a user cited as useful. From this listing, he could determine not only who had last written the file, but also which computer it came from (information which might be as useful to the user who had checked the file in as it would be to his collaborators).

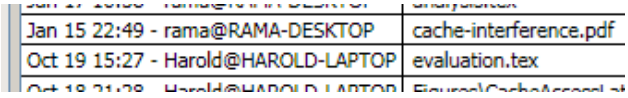

Figure 4. Snippet of file provenance information

Sync complete. When did syncing in each direction (inbound and outbound) complete? It is important that this information be unambiguous so a user can be confident that local changes have been fully propagated, and that incoming files from other computers are up-to-date. In other words, it is important to make sure the user knows that another computer was successfully contacted (sync began), and that the sync finished (sync complete). If the sync involves a cloud replica, users need to know the status of this connection that is assumed to be 'always on'.

Changes. Who has synced with the repository since the last time the user looked, and from where? What did they change, and have they added new files? This information provides useful feedback when collaborators are working at a distance (has everyone seen my changes yet? Has a specific co-author started to work on the paper yet?) and in a decentralized system, it reassures users that a remote collaborator's repository has synced with the others.

Some aspects of the sync process were visible and controllable, yet users did not seem to find them useful. For example, users were more apt to repeatedly force a sync than they were to change the sync interval to be more frequent (it was 5 minutes by default). We also found that users wanted to know when the cloud was available, and needed progress indicators for syncs that were slow to complete.

Collaborators in UC3 were sharing numerous large data files and presentations, in addition to the files directly relevant to the writing process, and the collaborators in UC9 created many smaller files as part of co-authoring a scholarly monograph over the better part of a year. These two collaborations revealed shortcomings in the initial synchronization feedback model. Specifically, both of these collaborations found it hard to tell whether their local files reflected what was in the cloud, and what had been added to the repository. For example, the collaborator who joined UC3 was not sure when the first sync was complete, or how long the sync would take. In an email, he said:

> *"After the first couple files showed up I thought the repository was somehow smaller than what [I] expected but then other files started showing up. Of course, being an impatient user, I have hit the sync button a few times so am [somewhat] unclear on whether that causes it to go retrieve additional files (via magic) or if they are being downloaded over some schedule so as to not swamp the network."*

Thus, the person who had set up the repository had to explicitly specify how many files to expect, and roughly what was in the repository. Would it have been easy to add such an indicator to Cimetric's user interface? This is a case in which the base technology did not support such a change: the sync protocol did not surface this information. Nor could either user control the sync order: files that were more germane to current tasks might well be the last to sync. It is frustrating for both sides of a dyad to watch a series of large (but not immediately necessary) files sync while the small file necessary to make progress on the current portion of the joint effort is waiting in the wings.

CONCLUSION

Through the development of a file synchronization application, and by observing its local use for scholarly collaboration, we set out to make three types of contributions: (1) to better understand the role of a cloud store in file synchronization; (2) to build a bridge between device synchronization and file sharing; and (3) to understand which aspects of synchronization to reveal to users.

What we learned was that a confluence of factors—organizational firewalls; the power-saving mechanisms and policies that are becoming increasingly commonplace; and fluid unanticipated configurations of people and computers—made it necessary to give users the option of syncing with the cloud. Yet there are reasons to keep a cloud replica optional, rather than making it a fixed element of every session. Performance, flexibility, cost, and

privacy all arose as reasons to retain the possibility of peer-to-peer syncing. For example, if users are working with highly sensitive material—e.g. code that represents significant intellectual property or data that might compromise study participant privacy—they may not want to store it on an external cloud service. Furthermore, peer-to-peer syncing was much faster than cloud-based syncing when the collaborators remained on the same subnet and were working in a semi-synchronous way as a deadline approached. Thus the topology-independent aspect of Cimetric was successful *as long as there was an option to create a cloud replica of working files or shared content*.

What of our attempt to separate device sync and file sharing? Did the bi-level design add needless complexity? Yes and no. The difficulty of understanding how to configure the system and the difficulty of changing configurations on the fly made users less likely to take advantage of bi-level syncing. Yet the work-arounds we observed (such as UC9's adoption of Dropbox to sync worksets) convinced us of two results: (1) *maintaining a distinction between the two types of syncing is useful* and (2) *worksets would have benefitted substantially from the option to sync with the cloud*.

Finally, we consider the sync information we made visible, and what we did not. In most situations, people do not examine when the last sync occurred, nor do they check the provenance of a file (where they received it from at sync time). Yet when breakdowns occurred, that information—and more (as was apparent in the feedback from UC3 and UC9)—was useful. Because we fielded the Cimetric application among technically-savvy users, it would be interesting to see whether the sync information would interpreted correctly among different user populations; some of it (when the last sync completed, who last edited a file, and from where) can be reassuring and possibly vital to interpreting what is going on.

Our investigation underscores the value of file synchronization in domains that stress the co-creation of content, just as it highlights some of the difficulties and pitfalls of sync applications. In the end, one question remains: are any of our collaborations still using Cimetric? As Grudin observed in his study of a collaborative writing tool [13], most of our collaborations did not continue using the tool after their specific activity had concluded, although several used it again when new writing tasks arose. This long-term use gives us hope that a cloud-optional approach that bridges between the rhythms of personal file sync and collaborative file sharing is a viable way to support content co-creation.

ACKNOWLEDGMENTS

We would like to thank our patient and daring users, especially those who used Cimetric in its early days.

REFERENCES

1. About Git. http://git-scm.com/about
2. Apache Subversion. http://subversion.apache.org/
3. Beck, E. & Bellotti, V. Informed Opportunism as Strategy: Supporting Coordination in Distributed Collab. Writing. *Proc. ECSCW'93* (1993), 233–248.
4. Belaramani, N., Dahlin, M., Gao, L., Nayate, A., Venkataramani, A., Yalagandula, P., & Zheng, J. PRACTI replication. *NSDI'06*, USENIX (2006), 59–72.
5. Bentley, R., Horstmann, T., & Trevor, J. The World Wide Web as enabling technology for CSCW: The case of BCSCW, *CSCW 6*, 2-3 (1997), 111-134.
6. Calder, B., Wang, J., Ogus, A., Nilakantan, N., Skjolsvold, A., McKelvie, S., Xu, Y., *et al*. Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency. *Proc. SOSP'11*, 143-157.
7. Cederqvist, P. *Version Management with CVS*. http://ximbiot.com/cvs/manual/
8. Dearman, D. & Pierce, J. It's on my other computer!: Computing with multiple devices. *CHI'08*, 767–776.
9. Dropbox. http://www.dropbox.com/ (retrieved 1 June 2012).
10. Farnham, S., Pedersen, E., & Kirkpatrick, R. Observation of Katrina/Rita Groove deployment. *Proc. ISCRAM'06*, 39–49.
11. Fitzpatrick, G., Marshall, P., & Phillips, A. CVS integration with notification and chat: lightweight software team collaboration. *CSCW'06*, 49-58.
12. Google Drive. https://drive.google.com/start (retrieved 1 June 2012).
13. Grudin, J. Groupware and social dynamics: Eight challenges for developers. *CACM 37*, 1 (1994), 92-105.
14. Karlson, A., Iqbal, S., Meyers, B., Ramos, G., Lee, K., & Tang, J. Mobile Taskflow in Context: A Screen Shot Study of Smartphone Usage, *CHI'10*, 2009-2018.
15. Kim, E. & Eklundh, K. How Academics Co-ordinate their Documentation Work, Royal Inst. Tech., Technical Report TRITA-NA-P9815, NADA, August 1998.
16. Kotla, R., Alvisi, L., & Dahlin, M. SafeStore: A Durable and Practical Storage System. *Proc. USENIX'07* (2007), 129-142.
17. Marshall, C.C. From Writing and Analysis to the Repository. *Proc. JCDL'08*, ACM Press (2008), 251-260.
18. Marshall, C.C. and Tang, J. That Syncing Feeling: Early user experiences with the cloud. *Proc. DIS'12*, ACM Press (2012).
19. McDonald, D., Weng, C., & Gennari, J. The multiple views of inter-organizational authoring. *Proc. CSCW'04*. ACM Press (2004), 564-573.
20. Müller, A., Rönnau, S., & Borghoff, U. A file-type sensitive, auto-versioning file system. *Proc. DocEng'10*, ACM Press (2010), 271-274.
21. Muller, M., Millen, D.R., & Feinberg, J. Patterns of usage in an enterprise file-sharing service: publicizing, discovering, and telling the news. *Proc. CHI '10*, ACM Press (2010), 763-766.
22. Noel, S., Robert, J-M. Empirical Study on Collaborative Writing: What Do Co-authors Do, Use, and Like? *Journal of CSCW 13*, 1 (2004), 63-89.
23. Posner, I. & Baecker, R. How People Write Together. In Baecker (ed.): *Readings in Groupware and CSCW*, Morgan Kaufmann (1993), 239–250.
24. Rader, E. Your, Mine and (Not) Ours: Social Influences on Group Information Respositories. *Proc CHI EA '09*, ACM Press (2009), 2095-2098.
25. Ramasubramanian, V., Rodeheffer, T., Terry, D.B., Walraed-Sullivan, M., Wobber, T., Marshall, C., & Vahdat, A. Cimbiosys: A platform for content-based partial replication. *Proc. NSDI'09*, USENIX (2009).

26. Salmon, B., Schlosser, S., Cranor, L.F., Ganger, G. Perspective: Semantic Data Management for the Home. *Proc. FAST '09*, USENIX (2009).
27. Sohn, T., Li, K., Griswold, W., Hollan, J. A Diary Study of Mobile Information Needs, *CHI'08*, ACM Press (2008), 433-442.
28. Schilit, B.N. and Sengupta. U. Device Ensembles. *IEEE Computer 37*, 12, (2004), 56–64.
29. Understanding Mercurial. http://mercurial.selenic.com/wiki/UnderstandingMercurial (retrieved 1 June 2012).
30. Voida, S., Edwards, W.K., Newman, M., Grinter, R., & Ducheneaut, N. Share and share alike: exploring the user interface affordances of file sharing. *CHI'06*, ACM Press (2006), 221-230.
31. Wang, Y., Gräther, W. & Prinz, W. Suitable notification intensity: the dynamic awareness system. *Proc. GROUP'07*, ACM Press (2007), 99-106.
32. Weng, C. & Gennari, J. Asynchronous collaborative writing through annotations. *Proc. CSCW'04*, ACM Press (2004), 578-581.
33. Whalen, T., Smetters, D., & Churchill, E. User experiences with sharing and access control. *Proc. CHI'06*, ACM Press (2006), 1517-1522.
34. Whalen, T., Toms, E., & Blustein, J. Information displays for managing shared files. In *Proc. CHiMiT'08*, ACM Press (2008).
35. Windows SkyDrive. http://windows.microsoft.com/skydrive/ (retrieved 1 June 2012).
36. Yamauchi, Y., Yokozawa, M., Shinohara, T. & Ishida, T. Collaboration with Lean Media: How Open-Source Software Succeeds Distance and Proximity. *Proc CSCW'00*, ACM Press (2000), 329-338.