TellTable:
A Server for Collaborative Office Applications

Andy Adler
School of Information Technology and Engineering,
University of Ottawa,
Ontario, Canada
adler@site.uottawa.ca

John C. Nash
School of Management,
University of Ottawa
Ontario, Canada
nashjc@uottawa.ca

Sylvie Noël
Communications Research Centre
Ottawa, Ontario, Canada
sylvie.noel@crc.ca

ABSTRACT

TellTable is a Linux-based application server which allows collaborative editing using single-user office productivity applications. Users log into the application via an SSL-enabled and Java-enabled web browser. Files to edit or view are opened in a VNC session on the server, the screen image is exported to a Java client in the user's browser, and all keyboard and mouse activity is then transmitted to the server. Editing conflicts are prevented using a locking protocol. In principle, TellTable allows any single user GUI application to be managed this way; we have tested it with OpenOffice, Microsoft Excel, and Gnumeric. Pilot tests showed that TellTable is usable over dialup and high-latency internet connections, but works best over higher speed connections. This paper reviews the technical design of TellTable and the interactions of the various components, as well as security issues.

Keywords: office suite, collaborative authoring

INTRODUCTION

Many work environments require collaborative writing and editing of documents, drawings, presentations and spreadsheets, ie. office-suite files; a good example is a scientific paper, in which researchers need to jointly develop and refine a document. Technology (telephone, email, and web tools) has simplified such collaboration. Currently, the typical way to collaboratively edit a document is to exchange draft versions between authors via email [8]. This means that control of document versions must be done by all members of the team. This introduces a significant additional burden on members, with the possibility of conflicting changes and missed contributions. One initial motivation for our work [7] that illustrates these difficulties is the common practice of managing course marks by emailing spreadsheet files between the various teaching assistants. The principal difficulty is that independent changes can be made to different versions, which must later be reconciled manually. It is also difficult to determine when a change was made, and why. Serious errors can be made very easily, such as the pasting of an entire list of marks over the wrong rows. A useful list of reported errors is maintained by the European Spreadsheets Risks Interest Group (http://www.eusprig.org/stories.htm).

Several software systems have been developed to address these issues ([7] presents a review of collaborative writing systems). In our opinion, the principal challenge with the design of such systems is ease of use. Team workers will resort to emailing private documents between each other if the collaborative system is not convenient to use. To address this and related issues, we have developed *TellTable*, a collaborative editing system designed to allow single-user office productivity applications (mostly word processing, spreadsheets, drawings and presentations) to be used in a collaborative framework. Advantages of such applications are the extensive effort that has been put into their usability design and the familiarity users already have with the interface. Users log onto TellTable via a web browser. Office productivity software is run on the server, and exported to the client's web browser via the VNC [11] protocol to a Java applet in the browser.

TellTable was initiated in 2002 as an approach for spreadsheet audit. We, with our colleague Neil Smith, had developed a tool to analyse the OpenOffice *calc* spreadsheet file change history to allow searching for particular patterns of activity of interest for audit applications [5]. However, it was clear that a user may make arbitrary changes to a file in their possession, including erasing or modifying the change history. In order to prevent such modifications, we began to develop a framework to allow *calc* to run on a server in such a way that the user has full access to the normal productivity functions, but without direct access to the files. It soon became clear that TellTable was a framework which could generally enable collaborative office software. So others can use, distribute, and modify TellTable, the server component of the project was licensed under the LGPL [3] in March 2004, and distributed from the URL http://telltable-s.sf.net in the Sourceforge repository. We are currently pursuing various enhancements to its functionality, and are actively interested in collaborating with others in its further development.

The rest of this paper describes the technical aspects of the TellTable software framework, including its interaction with each of the underlying software components. We review security issues in its implementation, and discuss tests of performance.

OPERATION

For the user, TellTable functions like a web application. The user enters the URL into an internet browser, and is presented with a login page. After entering a username and password combination, the user is presented with a screen

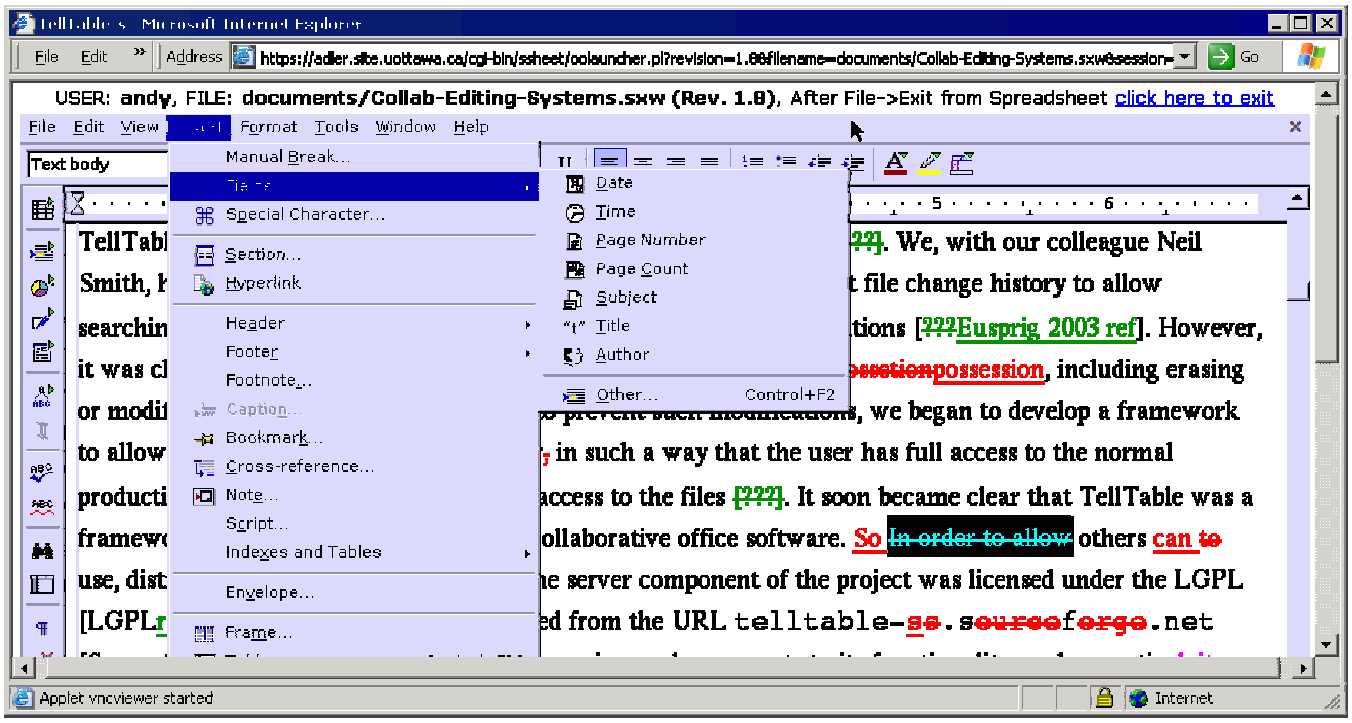

Figure 1: Screenshot of a TellTable document editing screen, showing a draft of this paper using the OpenOffice *write* software. The software is running on the web server, and the display is being exported into a Java applet in the web browser window. Full spreadsheet functionality is available, but with some security restrictions. A form below the Java applet offers the possibility of uploading new files or clipboard contents from the client machine to the server session.

showing the current status of all files to which they have access. At the lowest level of privilege, a user will be shown the file name, latest version number, as well as the date/time and user name of the last file edit. If another user is currently editing the file, it will be marked as "locked"; otherwise it will have a status of "normal". A user may choose to "View", "Download" or "Edit" a file (Edit is disallowed if the file is locked). Users with elevated privileges and who have appropriate software licenses may run other functions such as "Audit" or have access to the version history of files.

The "Download" option will cause the browser to download the selected file to the local machine. That file can then be manipulated locally, as desired, but such changes occur outside of the TellTable framework, and cannot be re-inserted into the file version history (without administrative privilege). “Download” is a more or less traditional browser function, and similar “buttons” can be added to TellTable. The other selections, namely "Edit", "View", or "Audit", send the browser to the application screen, which contains a Java VNC viewer applet. This applet connects to a VNC server that is running the appropriate software to implement the chosen function. For example, a choice of "Edit" results in the screen shown in Figure 1.

The selected file is opened with OpenOffice running on the server, and all keyboard and mouse activity from the user in the applet window is sent via the Java VNC client to the server and then to the Office Suite software, which will then update its screen output which is then sent to the browser. Since most users are familiar with Microsoft Office, and Office suite software in general, they typically find [1] using such software within a browser window to be familiar.

Because of the constraints of the Java applet security model [13], some operations function differently from their counterparts on a client workstation. First, the implementation of copy-and-paste requires that we work around Java applet security that prevents applets from interacting directly with the clipboard of the client machine. Second, the user needs to quit both the office software and the browser window.

The designers of VNC provided a special pop-up window function to help with copy and paste problems. A user pastes clipboard content from the client machine into the Java applet pop-up window, which sends its contents to the server, where the X clipboard is populated with its contents. We have chosen not to use this approach because: 1) it only works well with "traditional" X windows applications and clipboard, while applications such as OpenOffice maintain their own clipboards; and 2) the pop-up clipboard requires an extra (and somewhat unnatural) step to be taken. We felt that as long as we required an extra step, it should be possible to provide significantly enhanced functionality.

Our design for copy-and-paste allows selection from three sources of data: clipboard text, local files, and server files. In each case, the new data is copied to a read-only file

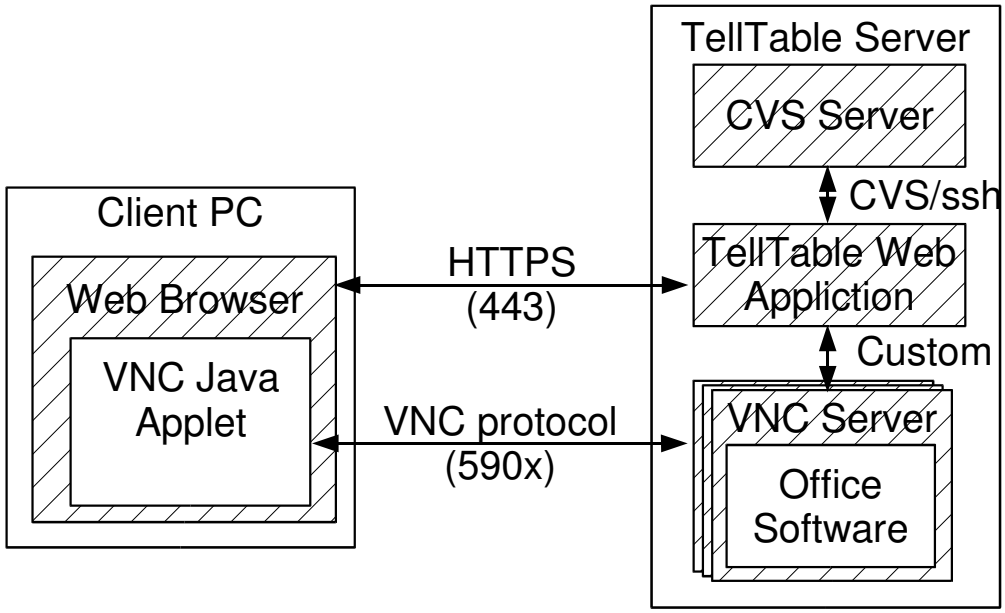

Figure 2: Block diagram of internetworked components for TellTable client and server system. The server consists of the components of a CVS service, web application service, and VNC services. These services communicate with each other via TCP/IP, and may be distributed onto several computers. The client browser communicates with the web application via https, and the VNC applet within the browser communicates with VNC servers via the VNC protocol.

on the server, and opened as a sub-window into the OpenOffice document. This allows the user to select text from the uploaded document and paste it into the working document as required.

TECHNICAL IMPLEMENTATION

In this section, we describe the technical implementation of the TellTable server. A block diagram of the intercommunicating components of the server and a client computer is shown in figure 2. Note that there is a semantic difficulty with the term "server" with X windows systems: the computer with the screen is the X server, while the application software runs on the X client. VNC, while building on top of X windows, reverses the semantics to the common usage: the VNC server is where the application runs. In this paper, we use these (familiar) semantics: the server runs the application to which the client PC (and screen) connects.

Client PC and Browser

TellTable is designed to offer cross platform support for client machines; tests have been performed with Windows 98/2000/XP, Linux, and Mac OS X client machines, using Internet Explorer 5.0+, Mozilla 1.4+, and Opera browsers. The client PC must have a graphical internet browser with support for the HTTPS protocol and Java applets. The size of the VNC applet is configurable at install time, and is currently set to 900 by 550 pixels. This selection works well with screen resolutions of 1024 by 768 or larger, but can be a little inconvenient to use for a PC with a smaller screen resolution setting.

VNC Server

The core of the TellTable server is a pool of VNC server processes, which may be distributed across several physical servers. Each server runs under a different low-privilege user id. Since there is no connection between VNC userids and TellTable client userids, we refer to these as pseudo-userids or *psuid*s. Additional security is inherent in TellTable in that the TellTable client userids are generally not registered users of the host Linux system. VNC servers are initialized at system boot time from a script `telltable-server` residing in `/etc/init.d`. Each VNC server has an associated X windows session, and a server port for VNC protocol access. At server startup, the VNC server also initiates a custom TellTable perl program `xstartup`. This program performs an initialization of the TellTable environment (verifying the existence of required files and directories), and then initializes a server for the TellTable commands. Thus *psuid* number 2 will have an X windows `DISPLAY=:2`, a VNC protocol server on port 5902 and TellTable server port on 5702.

Before executing each command, `xstartup` will delete and reset all software configuration files by unpacking them from a compressed archive. This serves to remove lists of previously edited files, and other modifications to the GUI menus from a previous editing session. Subsequently, `xstartup` listens for a command of the following form of the TellTable port:

```
{AUTHENTICATION_CODE} {COMMAND_STRING}
      {ACTION_STRING} {USERID}
```

An example string is

```
{7yr9Im4zhONNmDP+Cc0SPbSYdUw}
      {OOFFICE_OPEN} {sample.sxc} {andy}
```

`AUTHENTICATION_CODE` is a message authentication code (MAC) based on a code known to the web server and

VNC *psuid*. This code serves to authenticate the command string, and as a one-time password for the VNC server for this login session. `USERID` is the logged in userid of the client used to allow the editing software to record changes under the correct name. This is implemented for OpenOffice by modifying the appropriate XML configuration file. The `COMMAND_STRING` must match a predefined list of available commands. For example, the string "`OOFFICE_OPEN`" instructs `xstartup` to look for a file, specified by `ACTION_STRING`, in the input directory *`indir`*, move it to *`procdir`* and open it with the OpenOffice software. When the OpenOffice session is closed by the user, the file is moved to *`outdir`* (from which it is copied back to the repository as described below), after which the server listens for the next connection. Other values of `COMMAND_STRING` will open read-only clipboard contents into the currently open OpenOffice session, or will open other software, such as our spreadsheet audit tool.

Web Server

TellTable runs under the apache web server on Linux. The web server components are primarily composed of CGI script components written in Perl. There are no dependencies on advanced features of apache. To protect the security and login information for the VNC Java applet, TellTable must work with SSL encryption, such as apache-ssl (see http://www.apache-ssl.org), or the mod_ssl (http://www.modssl.org) extension to apache. TellTable maintains a database of system activity using the BerkeleyDB format [12]. The database is stored as a file on the web server, and is owned by the apache userid. Locking of the database between multiple CGI script invocations is implemented using the UNIX flock mechanism as implemented by the Perl module `DB_File::Lock`.

When a user logs into TellTable, the username and password presented are verified against the information recorded in the database. Subsequently, the file access screen is presented to the user. The list of directories the user is permitted to access is obtained from the database, and the lists of files in each permitted directory, as well as their current version, and the last user to edit, is obtained from the CVS repository. A user may also view the version history of a file, and may select "view" to open a previous version read-only in the VNC server. When a user selects a file to edit, or a previous version of file to view, the appropriate version is obtained from the CVS repository and sent to the VNC server, as explained in the next section.

CVS Repository

File versions are stored using CVS [2], which stores information using the RCS file format [14]. This format provides the capability to manage various advanced features of file versions, though at this time they are not used by TellTable. For example, CVS allows branching, merging, and file differencing. TellTable uses a simple sequential progression of version numbers, and uses CVS to allow extraction of older versions and to maintain descriptive text (logs) with each version. Because TellTable manages conflicts using locking, CVS capabilities for branching and merging are not required. Although file differencing would be of great benefit to users, the RCS file format was designed for plain-text files, and OpenOffice and other office software files are binary. This means that the "diff" functions of CVS (which calculate the differences between versions of text files) do not work. In general, a useful presentation of document differences would need to be determined at the application level, and some office software provides this function. Since the OpenOffice binary format is zip-compressed XML text, it seems that a form of difference presentation should be feasible, but we are not exploring this at the moment.

The TellTable web application interacts with the CVS repository to extract version information, to check out versions to view or edit, and to check in (or 'commit' using CVS terminology) newly edited versions. When a user chooses to edit a file, the Web server will check out the latest version from the CVS repository, copy it to the `indir` of the *psuid*, and send the appropriate command to the *psuid*'s server, which then opens the office software to edit or view the file. When the client quits the office software, the Web server tests whether the file is in the `outdir` of the *psuid*, and commits it to the CVS repository. CVS detects whether any modifications have been made, and, if so, adds the new version to the repository and increments the version number.

Security Details

This section reviews the features of TellTable designed to prevent legitimate users from performing actions which would normally require elevated privileges on the server. We assume that the server is configured to prevent arbitrary attacks from the internet. The TellTable server has a firewall configured such that only the HTTPS, and VNC protocol ports (590x) can be accessed from the internet.

At server initialization, the `xstartup` for each *psuid* has a different random code string embedded into it and also stored into the Web server database. When issuing a command, the TellTable server calculates a value `AUTHENTICATION_CODE`, an SHA1 based message authentication code (MAC) based on the command text and the stored code. This MAC is verified by `xstartup` before acting on any command. Commands with an invalid `AUTHENTICATION_CODE` are ignored with an error. We note that the current implementation is potentially vulnerable to a message replay attack which could be countered by using a challenge/response protocol. However, since the TellTable server port is not accessible from outside the server, we consider this issue to have a lower priority for improvement.

The `AUTHENTICATION_CODE` also serves as a one-time password for the VNC server. VNC authentication uses a challenge/response protocol based on the DES cipher. When a new VNC client connects to the server, a password value is entered which is tested against the value in the hashed password file `.vncpasswd`. When a command is sent to `xstartup`, the value of `.vncpasswd` is modified based on the `AUTHENTICATION_CODE`. At the same

time, when the web server creates the HTML page with the VNC applet, a corresponding password is created based on the code value. This mechanism ensures that the user of a previous VNC session cannot eavesdrop on a future session since the authentication information will no longer be valid. One source of some confusion to us was the fact that the VNC implementation of DES uses a permuted byte order compared to the standard one [6].

Each *psuid* runs as a member of the UNIX group `tt-group` and as a different low privilege user, `tt-uid####`, where `####` is the *psuid* number. Each *psuid* has processing directories `indir`, `procdir`, and `outdir`. `procdir` has permissions set to be private to the *psuid*, while the others can be read and written by `tt-group`. The userid under which the web server runs is configured to also be a member of `tt-group`, and is thus able to copy files to and from the *psuid*s. For TellTable configurations using separate computers for the web and VNC servers, the `indir` and `outdir` are shared with the web server by a network file system.

Perhaps the most important potential source of security vulnerabilities in TellTable is the client access to a VNC session. Although we only allow users to execute a small set of Office productivity applications, these applications are large, complicated, and not designed with this type of security in mind. It is therefore quite likely that a user may be able to obtain general shell access via the VNC session, for instance, by finding a way to get an Office application to launch shell commands. In this situation, the client user will have all the privileges of the *psuid* under which the VNC server runs. Given this possibility, we have designed TellTable to reduce the privilege of the *psuid*s as much as possible, so that a *psuid* cannot read or write web server, CVS service, or UNIX administrative and executable files. In order to further reduce the access privileges, we plan to run each *psuid* within a restricted directory using `chroot`, in which it has access only to required files and binaries.

When the CVS repository is on the same computer as the Web server, file versions are stored under the userid of the Web server. If the CVS repository is on a separate machine, then CVS commands are transported via SSH encryption. SSH authentication credentials are stored using the `ssh-agent` mechanism at Web server startup. This way, the authentication information is not available to the Web server.

IMPLEMENTATION AND TESTS

Currently, TellTable is implemented on a Dual Processor Linux server, running Debian based Xandros Linux. It is being used by several workers at the University of Ottawa, the Communications Research Centre of Canada, the University of Vienna, and by two independent teams of two people linking Ottawa and Cardiff and Ottawa and Toronto. Applications include collaborative writing of papers, course material, and maintenance of course marks and documentation. As mentioned in the introduction, the spreadsheet files are typically used to record course marks and represent a demanding test of a collaborative document system.

In the autumn of 2003, we performed a pilot study of TellTable for this application [1]. Results showed that users were largely appreciative of the features of the system (especially the ability to know one's changes would not be lost). Overall, usability was good. One concern that we had was that responsiveness would suffer on slow internet connections with high latency, such as dialup and connections from far away. We were pleasantly surprised to find that, although slightly slower, TellTable was quite usable in both situations. We attribute this to the efficiency of the VNC protocol design [11].

Some software bugs were triggered by patterns of usage of pilot users. For example, one interesting bug, and possible security issue, affected a user who would use their email account at `hotmail.com` to click on the link to the TellTable server that had been sent to them. However, hotmail opens URLs within a frameset that uses javascript to rewrite HTML to prevent "breaking out" of the hotmail frame by linking to the TellTable server via a hotmail server. Generally, the hotmail rewriting would incorrectly rewrite the VNC applet frame, rendering it non-functional. The solution was to require logging into TellTable from a new browser window. One concern is that a web mail service could use this technique to allow capture of passwords and other security information.

Our tests show TellTable has good scalability [4]. TellTable has the following memory requirements:

$$151 + 5.8(\text{available } psuids) + 17.0 * (\text{used } psuids) \text{ MB}$$

using OpenOffice *calc* in each session. On top of this requirement is the memory required for each open document. This result indicates that the server memory requirements are relatively small compared to that required by the operating system and the document file itself. Performance was calculated by performing simultaneous complex spreadsheet calculations in each *psuid*. Results show that the TellTable server evenly distributes available computational resources with very little overhead (~1%), suggesting that a moderately sized server (1GB memory) should be able to support 10-20 *psuid*s depending on the requirements of the users. Since most users of Office applications make sporadic use of computational power, it may be more efficient to use a powerful server such as TellTable with less powerful client PCs.

DISCUSSION

We have described the TellTable collaborative document editing system in terms of its technical design. We believe that TellTable is portable to other UNIX platforms, although we have no plans to do so. A port to Windows, however, would require a significant rewriting, as Windows does not easily allow multiple graphical sessions to run under different userids, as is required by TellTable. Perhaps the only advantage of the Windows platform is for running applications such as Microsoft Office. However, it is now possible, using CodeWeavers Crossover Office, to run Microsoft Office software on Linux. In preliminary tests, we were able to run Microsoft Excel remotely with TellTable.

We have considered using a faster framework for dynamic web content than CGI, such as mod_perl. However our current tests show that for reasonable loads (of up to ten simultaneous users) the speed of the web server does not significantly degrade. Indeed, most delays in the web server are spent interacting with other system tools, such as the CVS or VNC servers.

A possible annoyance with our design is the detection of "real" changes in files. For example, in Microsoft Word, opening a document, scrolling through and saving it, may result in a modified file. A version control system such as TellTable, will save these "versions", unless software were written to detect such "unchanged" files. For the moment we have chosen to wait and see if this is problematic.

Initially, we considered dynamically creating a new VNC session when requested by the user. This approach proved infeasible because VNC servers require significant time to start (5 sec.) resulting in additional delay for the user. Worse, when the VNC server shuts down, its TCP/IP connection is left in the TIME_WAIT state and cannot be restarted for up to two minutes. An approach based on dynamically started VNC sesssions would need to work around such timing considerations. Also, initiating VNC sessions for other userids requires elevated privilege for the web server, which may introduce security issues.

We also contemplated maintaining a VNC session for each physical system user. This requires no *psuid*s, making security analysis easier. Unfortunately, this approach would require a large memory and processor capability to support a large number of users. Furthermore, since each VNC server requires its own TCP port, it would require many open ports. As currently implemented, a VNC server is limited to 99 open sessions (ports 5901-5999). Load balancing with multiple servers is another difficulty. If all logged on users happened to be allocated to the same VNC server computer, then other machines would not be able to assist in supporting the computational load.

Our future technical work with TellTable involves expanding the set of applications we can launch and use with the infrastructure. We are particularly interested in tools like Gnumeric (www.gnumeric.org) that appear to be smaller and quicker to load than OpenOffice. We are also exploring ways to integrate work flow capabilities into the file-choice screen, since automation of the flow of files and information should enhance the utility of the infrastructure. There are also a number of housekeeping tasks for administering TellTable that are currently implemented as command-line tools, that could be much more user-friendly.

In conclusion, TellTable is a workable framework allowing single user office productivity applications to be used collaboratively. Moreover, this framework is open-source and runs on inexpensive hardware. The TellTable approach benefits from considerable effort put into the development of user-friendly features in large software packages, while not requiring a large effort to render these collaborative. Pilot results show that users are generally able to use their familiarity with such software packages to work easily with TellTable.

REFERENCES

1. Adler, A. and Nash, J C. (2004) Knowing what was done: uses of a spreadsheet log file *Spreadsheets in Education (eJSiE)*, 1(2):118-130, 2004. http://www.sie.bond.edu.au/articles/1.2/AdlerNash.pdf
2. Cederqvist, Per (2002). *Version management with CVS*, Bristol UK: Network Theory.
3. Free Software Foundation (1999). GNU Lesser General Public License, version 2.1, http://www.gnu.org/copyleft/lesser.html.
4. Nash, J. C., Adler, A. and Smith, N. (2004) TellTable Spreadsheet Audit: from technical possibility to operating prototype *Proc. 2004 EUSPRIG Conf. (European Spreadsheet Interest Group)*, (editors Patrick Cleary and David Ward), Klagenfurt, Austria, July 14-16, 2004, pp 45-56.
5. Nash, J.C., Smith, N., and Adler, A. (2003). Audit and change analysis of spreadsheets, *Proc. 2003 EUSPRIG Conf. (European Spreadsheet Interest Group)*, David Chadwick and David Ward, editors, 81~90.
6. National Bureau of Standards, Data Encryption Standard, FIPS-Pub 46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.
7. Noël, S., and Robert, J-M (2003). How the Web is used to support collaborative writing. *Behaviour & Information Technology* 22(4):245-262.
8. Noël, S., and Robert, J-M (2004) Empirical study on collaborative writing: What do co-authors do, use, and like? *Computer Supported Cooperative Work: The Journal of Collaborative Computing* 13 (1):63-89.
9. OpenOffice.org (undated) OpenOffice project api contents, http://api.openoffice.org/docs/DevelopersGuide/DevelopersGuide.htm
10. Organization for the Advancement of Structured Information Standards (OASIS) (2004) OASIS Open Office XML Format TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office
11. Richardson, T., Stafford-Fraser, Q., Wood, K.R, and Hopper, A. (1998). Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38. See also www.uk.research.att.com/vnc/, www.realvnc.com/vnc/, www.tightvnc.com/vnc/.
12. SleepyCat Software (apparently 2003) Berkeley DB Reference Guide, Version 4.2.52, http://www.sleepycat.com/docs/ref/toc.html
13. Sun Microsystems (2004) FAQ: Applet Security, http://java.sun.com/sfaq/
14. Tichy, W. F. (1985) RCS–A System for Version Control, *Software–Practice & Experience* 15(7):637-654.