Running Head: TellTable: Open Source Editing

Title: TellTable: An open source collaborative editing system

Authors:
Andy Adler
School of Information Technology and Engineering, University of Ottawa,
161 Louis Pasteur, Ottawa, Ontario, Canada, K1N 6N5
Tel: (613) 562-5800 ext. 6218, Fax: (613) 562-5175, Email: adler@site.uottawa.ca

John C. Nash
School of Management, University of Ottawa, School of Management
136 Jean-Jacques Lussier Street, Ottawa, ON, K1N 6N5 Canada
Tel: (613) 562-5800 ext. 4796, Email: nashjc@uottawa.ca

Sylvie Noël
Communications Research Centre Canada
3701, Carling Ave. PO Box 11490, Station H, Ottawa, Ontario K2H 8S2, Canada
Tel: (613) 990-4675 Fax: (613) 998-9648, Email: sylvie.noel@crc.ca

TellTable: an open source collaborative editing system

Andy Adler, John C. Nash and Sylvie Noël

Abstract: TellTable is a Linux-based application server which allows collaborative synchronous and asynchronous editing and viewing using single-user applications. Users log into the system 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, but users may allow others to synchronously edit a session. In principle, TellTable allows any single user application to be managed this way; it has been tested using the Microsoft and OpenOffice office suite and presentation software. Pilot tests show that TellTable is usable over dial-up and high-latency Internet connections, but works best over higher speed connections. This paper also describes the technical design of TellTable and the networked interactions and security design of each component.

Keywords: collaborative editing, collaborative infrastructure, groupware, system design, system security

1. INTRODUCTION

Many work environments require collaborative writing and editing of documents of all types (we use the term *document* for general content files). In simple cases there is essentially one author, with approvals and comments from others, while in other cases the document is genuinely collaboratively authored (Posner and Baecker, 1993). Such collaborative work can be asynchronous or synchronous (Kim and Severinson Eklundh, 1998). 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 (Noël and Robert, 2004). This means that control of document versions must be done by all members of the team, and introduces a significant additional burden on members, as well as the possibility of conflicting changes and missed contributions. For example, one initial motivation for our work (Adler and Nash, 2004) 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, and serious errors can be made easily.

Collaborative writing and editing solutions have been offered since the 1970s (see Newman and Newman, 1992, for a presentation of some of these early group writing systems, and Noël and Robert, 2002, 2003, for more recent systems), yet people prefer using single-user applications and distributing copies through email (Noël and Robert, 2004). More recently, designers have chosen to take advantage of the Web's popularity in the hopes of supplanting email's popularity. We see three writing and editing collaboration models offered on the Web:

1. The upload/download model: A web server is used as a document repository. Group members upland and download documents to this repository, but edit on their personal computer, using a single-user application. One example of this approach is BSCW[1] (Bentley et al., 1997a,b), although that solution also now offers online editing.

2. The web native model: A special editor is built from scratch to work through a web browser. Often (but not always) the resulting document is in HTML format. There are numerous available examples of this approach, including wikis (Leuf and Cunningham, 2001), SynchroEdit, Writely, WriteBoard, NumSum, iRows, and gOffice.

3. The hybrid model: A normally single-user application (e.g., OpenOffice, MS Office) is adapted to work through a web browser. Examples of this include our own TellTable (Adler and Nash, 2004; Adler, Nash and Noel, 2004, in print; Nash, Smith and Adler, 2003, 2004), coWord (Xia et al., 2004a), coPowerPoint(Xia et al. 2004b) and coStarOffce (Shen, Cheong and Sun, 2004).

Other non-web-based collaborative editing models include peer-to-peer and client-server models (e.g., MoonEdit, SubEthaEdit, Groove Networks). For more information on how TellTable compares to some of these other collaboration systems, see Adler, Nash and Noël (in print).

TellTable allows single-user productivity applications to be managed in a collaborative way. The underlying single-user software runs on a server; users log in with an Internet browser and view and interact with the software via a browser plug-in. This approach provides an easy migration path to users familiar with single-user applications, while benefiting from the significant development work in such productivity applications, because it runs those applications on a server that “wraps” the application in a collaboration environment. This is consistent with our philosophy that the principal challenge with the design of collaborative systems is ease of use – participants in a hard-to-use collaborative system will tend to resort to emailing private documents between each other.

TellTable was initiated in 2002 as an approach for spreadsheet audit; by running software on the server, all user interactions could be captured and centrally managed. Subsequently, TellTable was expanded into a framework for a general collaborative system. So others can use, distribute, and modify TellTable, the server component of the project was licensed under the GNU Lesser General Public License (Free Software Foundation, 1999) in March 2004, and is distributed from `http://telltable-s.sf.net`. We are currently pursuing various enhancements to its functionality, and are actively interested in collaborating with others on 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.

2. 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 showing the current status of all files to which they have access (Fig. 1). At the lowest level of privilege, a user will be shown the file name, latest version number, and date/time and user name of the last file edit. If the file is not in use, it will have a status "normal". If another user is currently editing the file, it will be marked as "Locked", unless the user has chosen to share the session in which case it will be marked as "Shared". A user may choose to "Edit", "View" or "Download" a file (Edit is disallowed if the file is locked). However, if the editing user chooses to "Share" a file, then the options available become "Shared-Edit" and "Shared-View". User privilege levels may be set to allow other functions such as "Audit", or to allow access to the version history of files. Figure 1 shows a view of the file access screen.

- INSERT FIGURE 1 APPROXIMATELY HERE -

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" and "View", 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 2.

The selected file in Figure 1 is opened with the appropriate application 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 software. The software interacts with user input and updates its screen output, which is then sent to the browser. Since most users are familiar with such software, they typically find using TellTable running such software within a browser window to be familiar (Adler and Nash, 2004).

Upon selecting "Edit" or "View" in Figure 1, the selected file is opened with the appropriate application 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 software. The software interacts with user input and updates its screen output, which is then sent to the browser. For example, a choice of "Edit" on the bottom file of Figure 1 results in the screen shown in Figure 2, where the file is opened by OpenOffice *write*.

By default, a user editing a file has exclusive write access to the file; this corresponds to an asynchronous workflow. In order to allow synchronous editing, a user editing a file may click "Share File" (Figure 2, top left). This will allow other users, with permissions to edit or view the file, the ability to simultaneously access the document. Since the underlying application is single-user, all TellTable users will see the same view of the document, and will have their keyboard and mouse interactions combined to the application. This means there is only one cursor so people must take turns to enter data.

While synchronous editing with shared input may appear to be a source of conflict, our experience is that such editing is normally done in conjunction with a conference call between participants. The group tends naturally to nominate one user to make the changes, while others watch and comment.

-INSERT FIGURE 2 APPROXIMATELY HERE-

Because of the constraints of the Java applet security model (Sun Microsystems, 2004), some operations function differently from their counterparts on a client workstation. First, the user needs to quit both the office software and the browser window. If the user closes only the browser window, the application is left running, and the user can access the application by logging back in. This can be useful, if a user wishes to switch the machine she is using, but can also allow an edit session to be abandoned. Abandoned sessions need to be detected (via an appropriate timeout) and automatically closed. Second, 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. The designers of VNC provided a special pop-up window function to circumvent this problem (Richardson, 1998). A user will paste clipboard content from the client machine into the Java applet pop-up window, which will then send 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 which function with the traditional X windows clipboard; advanced applications such as OpenOffice which maintain their own clipboards have difficulties interacting consistently with the X windows clipboard; 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 cut-and-paste places HTML input boxes, and allows input 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 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.

3. 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 3. A brief note on terminology: 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 (Richardson, 1998), 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.

- INSERT FIGURE 3 APPROXIMATELY HERE -

3.1 Client PC and Browser

TellTable is designed to offer cross platform support for client machines; tests have been successfully performed with Windows 98/2000/XP, Linux, and Mac OS X clients machines, using Internet Explorer 5.0+, Mozilla 1.4+, Firefox and Opera browsers. The client PC must have a graphical Internet browser with support for the HTTPS protocol and Java applets (Java version $\geq$ 1.0). The size of the VNC applet is configurable at install time, and is currently set to 900×550 pixels. This selection works well with screen resolutions of 1024×768 or larger, but can be a little inconvenient to use for a PC with a smaller screen resolution setting.

3.2 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 process 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–uids or *psuid*s. 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 to accept and process commands from the TellTable server. Thus *psuid* number 2 will have an X windows `DISPLAY=:2`, a VNC protocol server on port 5902 and TellTable server port on 5702. The firewall (described in more detail in section 4.2) only allows access on the VNC port.

The `xstartup` process running in each *psuid* accepts commands from the TellTable server to run single-user multimedia software on specified files with specified options (such as read-only). 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 on the TellTable server port:

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

An example string is

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

The `AUTHENTICATION_CODE` is a message authentication code (MAC) based on a SHA1 (National Bureau of Standards, 1995) message digest of the command text and a code known to the web server and VNC *psuid*. This code is randomly selected during TellTable installation. The `AUTHENTICATION_CODE` serves to authenticate the command string, and functions as a one-time password for the VNC server for the current login session. `USERID` is the logged-in userid of the client. It is used to allow the editing software to record changes and perform other customizations under the name of the user logged in to TellTable. Modifying the userid is implemented for OpenOffice by editing the appropriate XML configuration file within the *psuid*.

The `COMMAND_STRING` must match a predefined list of available commands. For example, "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.

3.3 TellTable 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) or the (Mod-ssl) extension to Apache. TellTable maintains a database of system activity using the BerkeleyDB format (Sleepycat Software). 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 hashed information recorded in the database. Subsequently, the file access screen (Fig. 1) is presented to the user. The list of directories the user is permitted to access is obtained from the database; the lists of files in each permitted directory, as well as their current version, and the last user to edit them, are 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.

3.4 CVS repository

File versions are stored using CVS (Cederqvist, 2002), which stores information using the RCS file format (Tichy, 1985). 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 multimedia and other office software files are typically binary. This means that the "diff" functions of CVS (which calculate the differences between versions of text files) do not work as expected. In general, a useful presentation of document differences would need to be determined at the application level, and some office software provides this function, but we are not working on interfacing to this capability 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.

4. SECURITY DESIGN

4.1 Security Threats

Collaborative work introduces many security concerns that do not exist for the individual author. We categorize these as server vulnerabilities, client computer vulnerabilities, access control and access level control.

Server vulnerabilities: collaborative systems require a networked server which, as a minimum, maintains a repository of file versions. Such network servers are potentially vulnerable to cracking through the applications running on the server. In more complex applications, the server is required to perform more complex operations, exposing it to a larger "pool" of possible attacks.

Client vulnerabilities: collaborative editing is not without risk to a client PC, even when exchanging draft versions via email. Most sophisticated office suites support macro languages, which are a popular vehicle for computer virus transmission. Since TellTable runs such applications on the server, it is important to protect against such macro viruses.

Access control: typically collaborative endeavours have a well defined group of participants. To ensure legitimate access, most current systems require a login with a username and password. There are well known problems with password based systems, largely from users forgetting passwords or inadvertently revealing them. This is a significant concern in a web-based system, such as TellTable. Also, a user logging into the system from an untrusted PC may inadvertently reveal access codes.

Access level control: group members often have different access privileges. For example, documents may be available to certain people on a "view-only" basis. Errors in setting access levels, especially in a complex system, can give users unwarranted additional privileges.

In the next section, we describe the design of TellTable to protect against such security vulnerabilities. We assume that regular security practices for an Internet accessible server are in place.

4.2 TellTable Security Design

The TellTable server has a firewall configured such that only the HTTPS, and VNC protocol ports (590x) can be accessed from the Internet and client machines. The firewall also completely blocks outgoing network access from TellTable machines. Thus, users and attackers will not have access to the X server or TellTable communication ports, or to any other server software running on the TellTable servers.

At server initialization, the `xstartup` for each *psuid* has unique 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`, a 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, we consider this issue to have a lower priority.

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 (National Bureau of Standards, 1977).

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 vulnerable aspect of TellTable is the VNC access to *psuid*s given to users. The concern is that users may be able to run arbitrary software as the *psuid* userid. This exposes the following vulnerabilities: users may "snoop" on other TellTable users; users may attempt to "hack" the server; or software may access the Internet to attack other machines or to act as a proxy or mail relay. In order to defend against such activities, TellTable has implemented three layers of defence. Firstly, we attempt to prevent such access by careful configuration of the single-user software running on TellTable; the options to browse the file system and run macros are disabled. However, since modern applications are very complex and powerful, we acknowledge that it is likely that bugs exist in such software which may permit users to circumvent these options, and to perform arbitrary actions as the *psuid* user. Secondly, to prevent arbitrary file system access, each *psuid* process is run within a file system compartment using the

UNIX `chroot` system call. This serves to isolate each *psuid* and the executable programs it is allowed to run from the rest of the TellTable file system. This, in turn, will help prevent "snooping" of the TellTable server and other TellTable users. Lastly, the firewall is configured to block all outgoing Internet activity, and only to permit incoming activity on VNC and HTTPS ports. This will help prevent use of TellTable for malicious Internet activity.

When the CVS repository is on the same computer as the Web server, then 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.

5. IMPLEMENTATION AND PERFORMANCE TESTS

Currently, TellTable is working on three different Linux servers 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 authoring of scientific presentations and articles, multimedia course material, and maintenance of course marks and documentation.

In the autumn of 2003, we performed our first pilot study of TellTable to manage spreadsheet files, which are typically used to record course marks (Adler and Nash, 2004). 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 dial-up 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 (Richardson, 1998).

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. 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. This example highlights the concern, however, that such a technique could be used to capture passwords and other security information.

In terms of scalability, our tests show TellTable works quite well (Nash, Adler and Smith 2004). TellTable has the following memory requirements:

$$151 + 5.8\times(\text{available } \textit{psuid}\text{s}) + 17.0\times(\text{used } \textit{psuid}\text{s}) \text{ MB}$$

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 spreadsheet data 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%). These results suggest 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 uses of Office applications make sporadic use of computational power, it may be more efficient to use a powerful server for TellTable with less powerful client PCs than individual powerful client machines.

6. DISCUSSION

We have described the TellTable collaborative authoring system in terms of its technical design and security aspects. TellTable runs on a Linux server and supports clients using most popular operating systems and Internet browsers. We believe that the TellTable server is portable to other UNIX platforms, although we have no plans to do so. A port of the server to Microsoft Windows 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" (ehttp://www.codeweavers.com), to run Microsoft Office software on

Linux. In preliminary tests, we were able to run Microsoft Powerpoint remotely with TellTable; however, it is unclear whether such use could be permitted by the software license.

We have considered using a faster framework for dynamic web content than CGI, such as mod_perl (http://perl.apache.org). 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 sessions 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.

Another possible design approach considered was to maintain a VNC session for each system user. This requires no *psuids*, 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 improved usability and security enhancements. One goal is to expand the set of applications we can launch and use with the infrastructure. We are also exploring ways to integrate workflow capabilities into the file-choice screen, since automation of the flow of files and information should enhance the utility of the infrastructure.

In conclusion, TellTable is a workable framework allowing single user 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. Its value is in making it relatively easy to make such software function in a collaborative way. Pilot results show that users are generally able to use their familiarity with such software packages to work easily and effectively with TellTable.

Appendix A – URIs for the web-based collaborative writing and editing systems

Multi-function or office suite systems:

BSCW: http://www.bscw.de/index_en.html

coStarOffice: (no web site available)

gOffice: http://www.goffice.com/

Groove Networks: http://www.groove.net/home/index.cfm

TellTable:http://www.telltable.com/

Authoring/writing/editing systems:

coWord: http://www.cit.gu.edu.au/~scz/projects/coword/

MoonEdit: http://www.moonedit.com/

SubEthaEdit: http://www.codingmonkeys.de/subethaedit/

SynchroEdit: http://www.synchroedit.com/

Wiki Wiki Web: http://c2.com/cgi/wiki?WikiWikiWeb

Writeboard: http://www.writeboard.com/

Writely: http://www.writely.com/

Slide presentation editing systems:

coPowerPoint: http://www.cit.gu.edu.au/~scz/projects/coppt/

Spreadsheets:

iRows: http://www.irows.com/xo/Welcome.do

Num Sum: http://numsum.com/

7. REFERENCES

Adler, A. and Nash, J. C. (2004): Knowing what was done: uses of a spreadsheet log file. *Spreadsheets in Education (eJSiE)*, vol. 1, no. 2, pp.118-130. http://www.sie.bond.edu.au/articles/1.2/AdlerNash.pdf

Adler, A., Nash, J. and Noël, S. (2004): TellTable: A Server for Collaborative Office Applications. In D.Li (ed.) *Proceedings of the Sixth International Workshop on Collaborative Editing Systems, Chicago, USA, November 6, 2004*. Published online, special issue of IEEE Distributed Systems Online http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc67 16bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/topics/collaborative/events/ iwces- 6&file=index.xml&xsl=article.xsl&;jsessionid=G82Zhlmn2yjY092TCYBLnmypqJqy QXXXBTnnddV5BFTmNLdTk5SW!554494575

Adler, A., Nash, J. and Noël, S. (in print): Evaluating and implementing a collaborative office document system. To appear in *Interacting with Computers*.

Apache-SSL, http://www.apache-ssl.org, Accessed: 2005-06-15

Bentley, R., Horstmann, T. and Trevor, J. (1997a): The World Wide Web as enabling technology for CSCW: The case of BSCW. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, vol. 6, issue 2-3, pp. 111-134.

Bentley, R., Appelt, W., Busbach, U., Hinrichs, E., Kerr, D., Sikkel, K., Trevor, J. and Woetzel, G. (1997b): Basic support for cooperative work on the World Wide Web. *International Journal of Human Computer Sudies*, vol. 46, issue 6, pp. 827-846.

Cederqvist, Per (2002): *Version management with CVS*, Bristol UK: Network Theory.

Free Software Foundation (1999): GNU Lesser General Public License version 2.1, http://www.gnu.org/copyleft/lesser.html.

Kim, E. and Severinson Eklundh, K. (1998): *How academics co-ordinate their documentation work and communicate about reviewing in collaborative writing*. Report #TRITA-NA-P9815, IPLab-151. KTH-Sweden.

Leuf, B. and Cunnigham, W. (2001): *The Wiki Way: Quick Collaboration*. Addison-Wesley.

Mod-SSL. http://www.mod-ssl.org Accessed: 2005-06-15

Nash, J.C., Smith, N. and Adler, A. (2003); Audit and change analysis of spread sheets. In D. Chadwick and D. Ward (eds.): *Proceedings of the 2003 EUSPRIG Conference (European Spreadsheet Interest Group)*, , pp. 81–88.

Nash, J.C., Adler, A and Smith, N. (2004): TellTable Spreadsheet Audit: from technical possibility to operating prototype. In D. Chadwick and D. Ward (eds.): *Proceedings of the 2004 EUSPRIG Conference (European Spreadsheet Interest Group)*, pp. 45-56.

National Bureau of Standards (1977): *Data Encryption Standard*, FIPS-Pub 46. National Bureau of Standards, U.S. Department of Commerce.

National Bureau of Standards (1995): *Secure Hash Standard*, FIPS-Pub 180-1. National Bureau of Standards, U.S. Department of Commerce.

Newman, J. and Newman, R. (1992): Three modes of collaborative authoring. In P.O. Holt and N. Williams (Eds.), *Computers and Writing: State of the Art.* Oxford: Intellect Books, pp. 20-28.

Noël, S. and Robert, J.-M. (2002): Assister l'écriture collective: Solutions sur réseaux locaux ou étendus et sur le Web. *Revue d'interactions homme-machine*, vol. 3, no. 2,pp. 95-114.

Noël, S. and Robert, J.-M.(2003): How the Web is used to support collaborative writing.

Behaviour & Information Technology, vol. 22, no. 4, pp. 245-262.

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*, vol. 13, pp. 63-89.

Posner, I.R. And Baecker, R.M. (1993): How people write together. In R.M. Baecker (Ed.): *Readings in Groupware and Computer-Supported Cooperative Work: Assisting Human-Human Collaboration*. San Mateo, CA: Morgan Kaufman, pp. 239-250.

Richardson, T., Stafford-Fraser, Q., Wood, K.R, and Hopper, A. (1998): Virtual Network Computing. *IEEE Internet Computing*, vol. 2, no.1, pp.33–38. See also www.uk.research.att.com/vnc/, www.realvnc.com/vnc/, www.tightvnc.com/vnc/.

Shen, H., Cheong, C.T. And Sun, C. (2004):CoStarOffice: Towards a flexible platform independence collaborative office system. . In D.Li (ed.) *Proceedings of the 6thSixth Iinternational. Workshop on Collaborative Editing Systems, Chicago, USA,* (CSCW 2004), *Nov.ember 6, Chicago, USA 2004*. Published online, special issue of IEEE Distributed Systems Online http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/topics/collaborative/events/iwces-6&file=index.xml&xsl=article.xsl&;jsessionid=G82Zhlmn2yjY092TCYBLnmypqJqyQXXXBTnnddV5BFTmNLdTk5SW!554494575

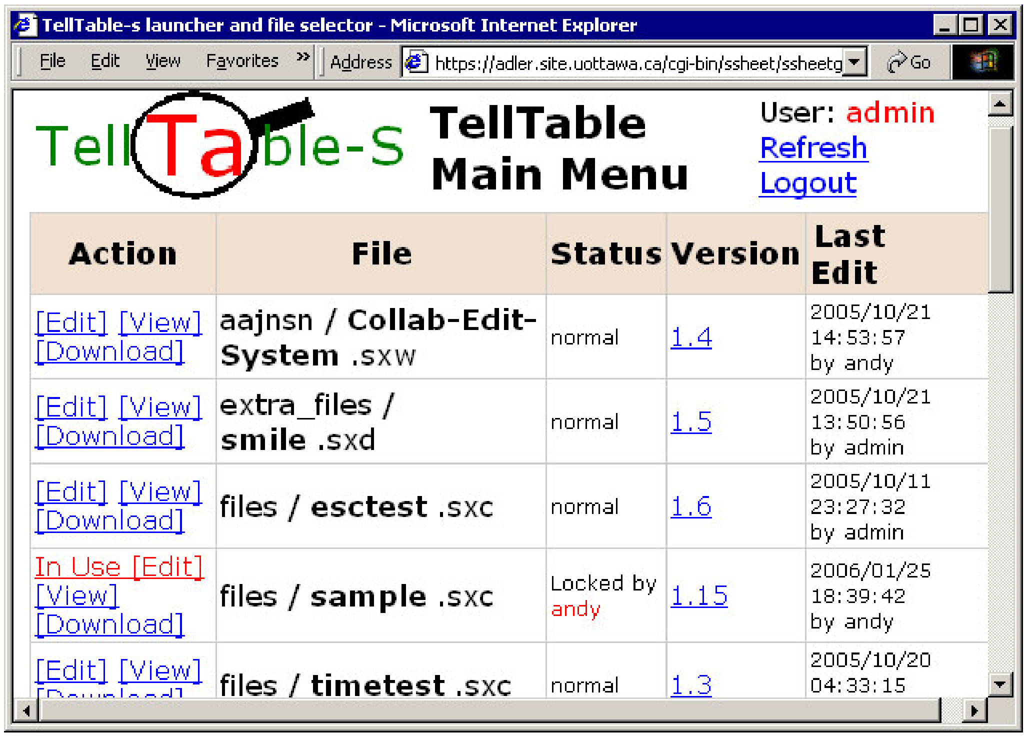
Sleepycat Software: *Berkeley Database*, http://www.sleepycat.com/, Accessed: 2005-Jun-15

Sun Microsystems (2004): FAQ: Applet Security, http://java.sun.com/sfaq/, Accessed: 2005-Jun-15

Tichy, W.F. (1985): RCS−A System for Version Control, *Software−Practice & Experience,* vol.15, no.7, pp.637-654.

Xia, S., Sun, D., Sun, C., Chen, D. and Shen, H. (2004a:. Leveraging single-user applications for multi-user collaboration: the CoWord approach.In J. Herbsleb and G. Olson (eds.): *Proceedings of ACM 2004 Conference on Computer Supported Cooperative Work, November 6-10, Chicago, IL USA*. New York: ACM Press, pp. 162-171.

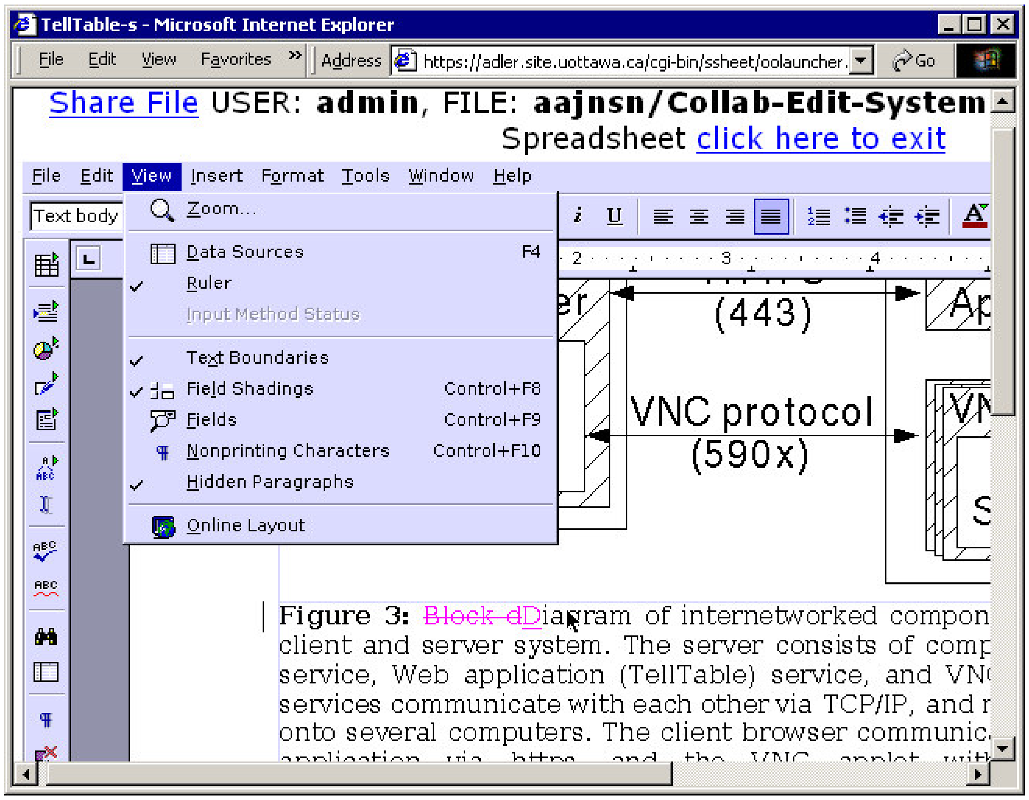
Xia, S., Sun, D., Sun, C., Chen, D. and Shi, Y. (2004b). Supporting interactive presentations with coPowerPoint. *S*. In D.Li (ed.) *Proceedings of the 6thSixth Iinternational. Workshop on Collaborative Editing Systems, Chicago, USA,* (CSCW 2004), *Nov.ember 6, Chicago, USA 2004*. Published online, special issue of IEEE Distributed Systems Online http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/topics/collaborative/events/iwces-6&file=index.xml&xsl=article.xsl&;jsessionid=G82Zhlmn2yjY092TCYBLnmypqJqyQXXXBTnnddV5BFTmNLdTk5SW!554494575


TellTable-s launcher and file selector - Microsoft Internet Explorer
File Edit View Favorites
Address https://adler.site.uottawa.ca/cgi-bin/ssheet/ssheetg
Go
TellTable-S

TellTable Main Menu

User: admin
Refresh
Logout

Action	File	Status	Version	Last Edit
[Edit] [View] [Download]	aajnsn / **Collab-Edit-System** .sxw	normal	1.4	2005/10/21 14:53:57 by andy
[Edit] [View] [Download]	extra_files / **smile** .sxd	normal	1.5	2005/10/21 13:50:56 by admin
[Edit] [View] [Download]	files / **esctest** .sxc	normal	1.6	2005/10/11 23:27:32 by admin
In Use [Edit] [View] [Download]	files / **sample** .sxc	Locked by andy	1.15	2006/01/25 18:39:42 by andy
[Edit] [View]	files / **timetest** .sxc	normal	1.3	2005/10/20 04:33:15


TellTable-s - Microsoft Internet Explorer
File Edit View Favorites Address https://adler.site.uottawa.ca/cgi-bin/ssheet/oolauncher. Go
File Edit View Insert Format Tools Window Help
Text body
Zoom...
Data Sources F4
Ruler
Input Method Status
Text Boundaries
Field Shadings Control+F8
Fields Control+F9
Nonprinting Characters Control+F10
Hidden Paragraphs
Online Layout
(443)
VNC protocol
(590x)

Share File USER: **admin**, FILE: **aajnsn/Collab-Edit-System** Spreadsheet click here to exit

Figure 3: ~~Block d~~Diagram of internetworked compon
client and server system. The server consists of comp
service, Web application (TellTable) service, and VN
services communicate with each other via TCP/IP, and r
onto several computers. The client browser communic

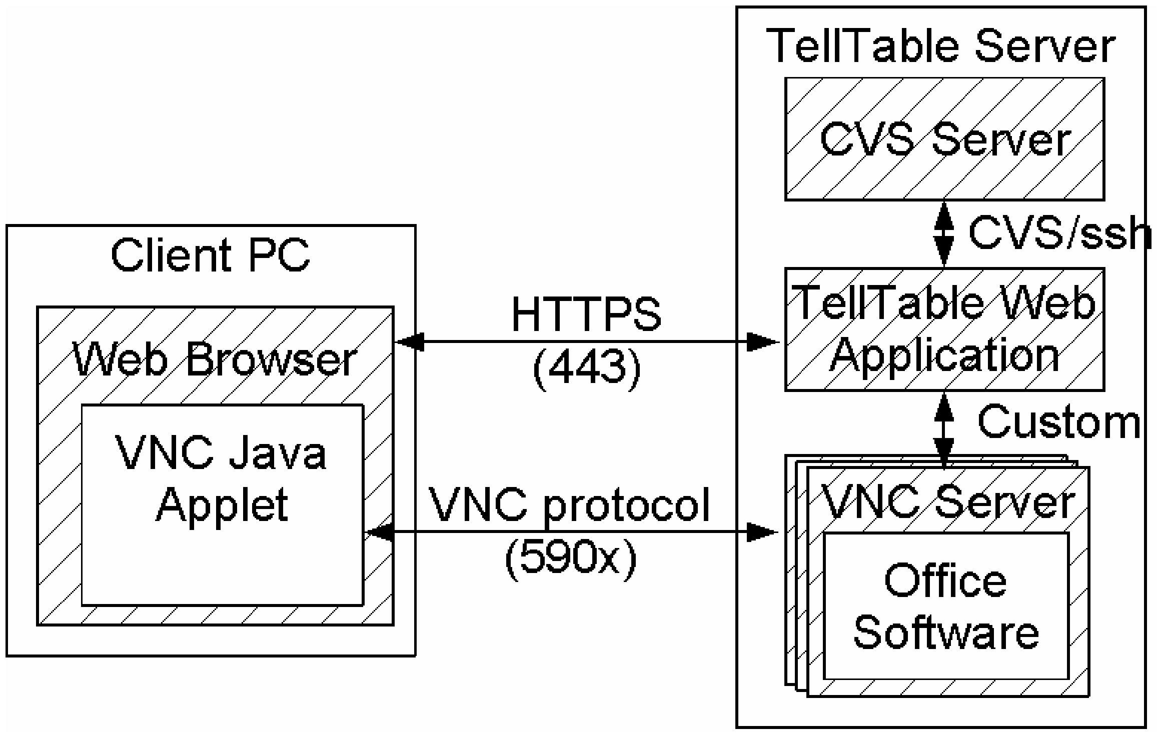

TellTable Server
CVS Server
CVS/ssh
Client PC
HTTPS
(443)
TellTable Web Application
Web Browser
Custom
VNC Java Applet
VNC protocol
(590x)
VNC Server
Office Software

Figure 1: TellTable document selection screen. Users are presented the option to Edit, View, or Download files. The file "sample" is currently in use, and is locked against editing by another user. The file "TellTable-s-logo" is also in use, but the editing user has chosen to share the file (by clicking "Share File" at the top left of Fig. 2). In this case, the options available to the user are "Shared-Edit" and "Shared-View". Clicking on these links will open a shared editing session, in which all users view the same application screen. The "Ver." shows the file version. Clicking on the version will present a file history and allow the user to view previous saved document versions. "Last Edit" is the time and userid of the last user to save the file.

Figure 2: Screenshot of a TellTable editing screen, showing a draft of a presentation compiled in the OpenOffice *presents* 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 software 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. Clicking on *Share File* will allow other users to share this editing session.

Figure 3: Block diagram of internetworked components for TellTable client and server system. The server consists of components for a CVS service, Web application (TellTable) 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.

Endnotes

---

[1] Appendix A presents the URIs for the various systems presented in this section.