Evaluating an instant messaging protocol for digital whiteboard applications

Lutz Gericke
Hasso-Plattner-Institut Potsdam
Prof. Dr. Helmert Str. 2-3
14482 Potsdam, Germany
Email: lutz.gericke@hpi.uni-potsdam.de

Christoph Meinel
Hasso-Plattner-Institut Potsdam
Prof. Dr. Helmert Str. 2-3
14482 Potsdam, Germany
Email: meinel@hpi.uni-potsdam.de

***Abstract*— Many whiteboard applications have been developed during the recent years. In distributed settings, where whiteboards have to be synchronized, proprietary protocols are often used for communication. Using reliable and well-known open standards can ease integration with existing systems. After evaluating existing instant messaging protocols, we look at applying those well-established standards for the problem domain of distributed whiteboards. The presented system does not only support synchronous working modes, but also asynchronous, by consequently capturing and analyzing communication data. The presented approach goes beyond existing systems in unifying those two working modes into one system. Evaluation of this in-situ capturing shows only little influence on the system's performance and therefore being an approach generally applicable to other problem domains.**

***Index Terms*— digital whiteboard, XMPP, instant messaging protocol, performance evaluation**

I. Introduction

Remote collaboration tools are well-established in today's companies. They support the communication over distances using audio/video conferencing, messaging, whiteboards, and desktop sharing. Often, this collaboration is limited to information exchange and communication can only hardly be retraced. Together with problems such as time shift over different continents, synchronous working modes are quite limited. Due to this limitations, regular face-to-face meetings are often the method of choice for successful collaboration. The result is less joint work and more reporting and documentation via e-mails and other documents about the efforts at each location. As we learned from interviews with employees of global companies, documentation is time-consuming and frustrating for all involved parties and often only done because project management demands it.

Besides these difficulties, e-mails and text documents often are not suitable to convey ideas and concepts that people have during their project work. It is very hard to communicate and understand why people took certain decisions and which were their most important concerns. Especially with creative work, which involves a lot of unforeseen ways of working, thinking about ideas or innovations and visualizations of concepts and designs, it is very difficult to write down the results of a meeting. Teams who are applying methods such as Design Thinking [1] often work with whiteboards, sketches and sticky notes and only in the end they "translate" their work to text documents or presentations.

A preferred tool for co-located work is the whiteboard. People standing in front of it can form a common ground of understanding by sketching their ideas, writing down notes, brainstorming on problem solutions, or simply working on shared todo-lists. A major benefit of a traditional whiteboard is its extremely user-centered design adopting input modes that are being used for thousands of years. Even children can easily work with this tool due to the simplicity of the used tools - pens, eraser and sticky notes. That is supposably one of the reasons for the large adoption of whiteboards in companies. On the other hand, whiteboards are still only rarely used in distributed settings. Some remote collaboration tools such as Adobe Connect support whiteboard sharing, but people refuse to use it. On a traditional whiteboard, users have to actively erase all the content from a board, whereas in known electronic whiteboard tools, whiteboards are often automatically erased, when people leave the communication channel. Moreover, the direct link between whiteboard surface and video conference is not not given, so that it is impossible to point on certain sketches on the whiteboard.

This leads to the main question of our research: How can we support people in their co-located as well as their distributed work using digital whiteboards? Can we bridge the (time)gap between continents and support collaboration more efficiently using digital communication channels?

Many whiteboard applications have been developed during the recent years. For distributed settings, where whiteboards have to be synchronized, proprietary protocols are often used for communication. Our approach uses an open standard for whiteboard data transmission, in order to benefit from existing tools as well as being able to easily integrate our system into existing solutions and reuse the generated data.

In the following, we elaborate on existing tools for whiteboard interaction as well as protocols that are being used in the domain of instant messaging. Our concrete communication protocol is introduced in order to understand the system architecture of the implemented system Tele-Board. Afterwards, we evaluate the appropriateness of an instant messaging protocol for digital whiteboards and show how the support for asynchronicity effects the system's performance.

II. Related work

Computer supported collaborative work (CSCW) is a field of research which brought up many different research projects and products on the market supporting distributed work. A typical schema to differentiate between those different solutions is the CSCW matrix shown in Fig. 1 (cf. [2], [3], [4]). The cells of this grid cannot be clearly separated, however many projects have a strong focus on one of them.

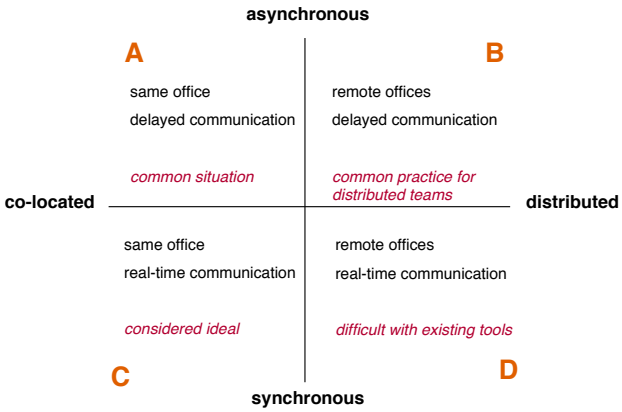


Fig. 1. Categorization matrix of working modes in CSCW

Older approaches such as *ClearBoard* [5] or *VideoWhiteboard* [6] are distributed over short ranges, but do not care about network communication over large distances. Gestures and interaction on the board are more in the focus of their work, so that it is impossible to edit remotely created artifacts.

Designer's outpost [7] is one of the more recent approaches in the area, but focusses on image vision methods and interaction concepts. A history is also provided, but users have to explicitly save certain states, which are then stored as images in the history. Editing of remote content is also not possible.

The general approach of using an instant messaging protocol for whiteboard interaction purposes is not completely new, but it still lacks tool support. There is one application called *Coccinella* [8], which essentially is an XMPP chat client with an added drawing surface. The development seams to be discontinued since two years and it also lacks any asynchronous features. The used approach here is a combination of SVG and XMPP as it is also described in [9] and [10]. There have been some proposals in order to standardize this protocol, but since 2006 there is no new progress in this area. XMPP itself is the most widely used open protocol for instant messaging applications.

Summing up, the problem domain of whiteboard interaction is a field of research, which brought up many challenges relating human computer interaction or image synthesis such as *Thor* [11], that is why the network communication infrastructure is not in the focus of the work. Often, proprietary protocols are used for communicating between the multiple locations. Recording is also done but more on a representation level (video streams or images) and not on a artifact level (cf. [7]).

Our approach should benefit from open standards in the field of internet communication protocols as well as being open for easy capturing and archiving of recent whiteboard sessions, which enables fast analyses and effortless pause and resume behavior including implicit saving of the communication history.

III. Communication protocol

There is a large variety of input devices and thus highly different requirements for the communication protocol concerning the connected devices. Generally there is a differentiation between Tele-Board-aware devices being equipped with a special client software and those coming out of the box that can participate in the design session with an existing client software that uses the communication protocol. It should be possible to write and send sticky notes from every Internet-enabled device without developing a special client software for it. When using special capabilities such as drawing, it can be acceptable to develop an adapted application for that platform.

Every single component has very special needs in terms of user interface development, data structures, and communication methods. An important decision was using the Extensible Messaging and Presence Protocol (XMPP) (defined in RFC 3921, cf. [12]) as a communication protocol in order to support a variety of input devices and different platforms. There are several implementations on almost every platform and it is supported by a many existing instant messaging clients.

XMPP is an open standard and is typically used as a chat and instant messaging protocol. Over time it has been extended to support voice, video, and file transfer. XMPP (formerly known as Jabber) is used in several instant messaging tools such as Google Talk [13] or Psi [14]. The communication is build upon a client-server model. Authorization, session and roster handling is managed by the server. People can connect with every possible client without transferring any configuration from client to client except for username and password.

The *Server-Buddy* plugin, which is deployed into the Openfire[1] server, acts as a so-called PacketInterceptor to read all messages sent between whiteboards in order to archive them in a database (see figure 2). Special packets such as a request-message for resuming a session are filtered out, directly answered and not stored in the whiteboard history. This procedure makes the collected information usable in the history component, so that every state of a Panel can easily be reconstructed.

Technically, all communication is routed over the XMPP server. In terms of XMPP, the whiteboards chat with each other. The used method for communication between any two partners is a multi user chat (MUC), as it is defined in the XMPP specification. One whiteboard session is reflected in a MUC room on the server, which is automatically created for each new session. The body of the messages is extended to

[1]Openfire is an open source XMPP server software, http://www.igniterealtime.org/

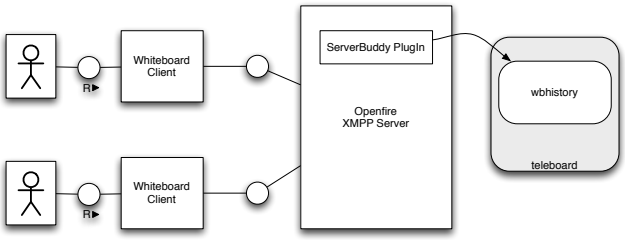


Fig. 2. Communication interception between the whiteboard clients

support the needs of the Tele-Board synchronization. XMPP-Clients producing text-based sticky notes, direct their messages to this room, the server plugin changes the packet to form a valid sticky note description.

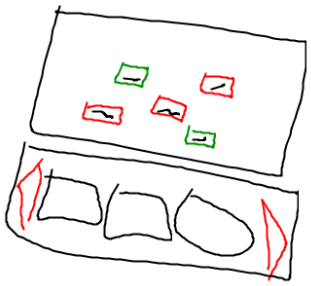
There is an operation code signaling the type of message stored in the message body. There are multiple types of operation codes (opcodes) used:

- `WHITEBOARD_SYNC_NEW` - an element is newly created
- `WHITEBOARD_SYNC_CHANGE` - an element is changed
- `WHITEBOARD_SYNC_DELETE` - an element is deleted
- `WHITEBOARD_SYNC_ALL` - request to send all whiteboard content to the communication partner
- `WHITEBOARD_SYNC_ALL_ANSWER` - answer to a sync-all request, contains the whiteboard content
- `WHITEBOARD_SYNC_PICK` - putting an element into the server-wide clipboard
- `WHITEBOARD_GET_CLIPBOARD` - request clipboard content list
- `WHITEBOARD_GET_CLIPBOARD_SET` - request clipboard content for specific id
- `WHITEBOARD_SET_META` - request to store meta-data on the server
- `WHITEBOARD_DELTA` - special operation for fast updates on single variables (e.g. position updates during drag operations)

There are multiple devices with different capabilities concerning the handling of the XMPP messages. Messages using the mentioned opcodes are sent between the whiteboard clients to synchronize the whiteboard content. The peripherals will send the message content to the local whiteboard. From the Whiteboard Client the content will be synchronized to the remote whiteboard locations. The `WHITEBOARD_SYNC_*`-messages are important for the whiteboard history and will be handled in the Server-Buddy plugin to build up the historical data. `NEW`, `CHANGE`, `DELETE` will be directly archived, `ALL` and `ALL_ANSWER` will not appear in the history.

XMPP as a communication language between the clients turned out to be very appropriate. The development of the whiteboard clients can rely on a sophisticated infrastructure e.g. for user handling and message routing. It did not have to be implemented from scratch, but could be used as an existing part of the protocol. A synchronous whiteboard session can be established without any server changes. What will not work without the server-side plugin, is the archiving of a session and the restoring of an earlier state of the whiteboard session, which means the asynchronous features of the system.

The payload of the chat messages exchanged between the whiteboard clients is an XML-encoded text representation of a single whiteboard element. The example in Fig. 3 shows a set of scribbles, having different properties; the XML extract, shows one single scribble element. There are several attributes describing the element in order to reproduce it at the remote location: `x` and `y` describe the position, `strokecolor` the color of the path, and `d` represents the path itself in SVG-notation. SVG was chosen because it is an established standard and can be directly used for screenshot rendering with only little string conversions.



```
<path id="lutz@fb10dtools_654"
    strokecolor="0.0,0.0,0.0"
    d="M 2286.0 1237.6575 L 2283.0..."
    x="119.0" y="354.0"/>
```

Fig. 3. XML representation of one single Scribble vs. graphical representation of multiple Scribbles with different colors, paths, and locations

IV. Tele-Board - A Whiteboard Application for Synchronous and Asynchronous Settings

We developed the whiteboard software suite Tele-Board. First of all, it supports synchronous working modes. That is, people having the possibility to collaborate over distances or locally at the same time. The main idea is to plug the whiteboards together. XMPP names the notion of bringing together multiple participants as a group chat or MUC (multi-user chat). One message is send to the server and from there distributed to all connected clients.

To realize the specified functionality for asynchronous features in a software system we developed three main functional units:

- interception of message flow
- storage of communication data
- enabling interaction with the history data in an appropriate user interface

The communication should be captured on-the-fly, which has influenced the selection of the technology insofar as it must be possible to analyze packets separated from the message routing. Central roles in the overall system are represented by the message server and its plugin architecture, the web-based management system, and the database management system. The history functionality is a concept that is implemented

as a cross-cutting concern in all parts of the system. It can not be realized as one single component, because it enriches the functionality of the other components. A client-server architecture is used for synchronizing the participating whiteboards. A central history archive located at the server is more suitable than e.g. a peer-to-peer model as it is used in [15], because all statistical data should be kept together in order to be analyzed conveniently and enable asynchronous work.

To understand where the communication takes place and what should be captured, the existing communication infrastructure should be briefly outlined: A *Workspace Hub* is a computer system running the whiteboard client software and is located at a digital design space. This computer combines all physical components at this location. Several input devices are paired to it. Cameras make it possible to also synchronize a video stream of the people standing in front of the whiteboards in order to provide a communication channel supporting eye-contact and gestures regarding the whiteboard elements (e.g. pointing on a sticky note). How the multiple Workspace Hubs work together and how they communicate with the help of the server is shown in Fig. 4.

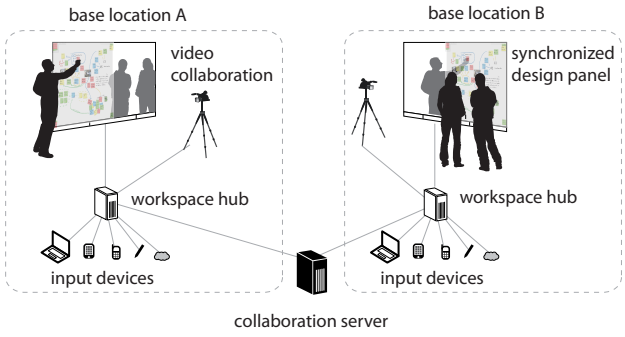


Fig. 4. Overall setup of the communication infrastructure centered around the workspace hubs that are connected via an XMPP server

Projects and *Panels* are important concepts in this context. A Panel p describes the sequence of events e_n executed on one whiteboard in temporal order of these events ($p = (e_1, e_2, e_3, ...)$). An event is a tuple of attributes describing which action has happened where, by whom, and when, to keep the temporal order of the events. Each event has an action code, which can be NEW, $CHANGE$ or $DELETE$ to describe the event type. A Project pro is the collection of multiple Panels ($pro = \{p_1, p_2, ...\}$) in order to configure them with rights to edit/view/delete.

The architecture outlined so far is reflected in three major components:

- a web portal for creating, viewing, and changing projects and panels
- a history browser for exploring earlier states of a whiteboard session
- a whiteboard client, for interaction with the system

The current Tele-Board system can be run on every computer equipped with a browser and a Java Runtime Environment. Nothing has to be installed on the computer. As a starting point, the user logs into the web portal using username and password. When successfully logged in, it is possible to browse through projects and panels, edit them or create new panels or projects. Every whiteboard session (panel) has a history, which is explorable also within the web portal. The tool used for this history interaction is called *history browser*. With the help of this tool, the user is able to scroll through the timeline of a session. This is possible, because every single event is stored in the history archive. Thereby, screenshots of every second in the editing process can be rendered by the server. When a user finds an interesting state he wants to continue, he can choose to start from this point by branching into a new parallel panel or resume the work from the end of a whiteboard session. Resuming the work means, starting the whiteboard client directly from the browser via Java Webstart. The client requests the latest state of a session via XMPP so that a user can continue working with the most recent information on the whiteboard.

Having this architecture in mind, the question comes up, how efficient this selection of protocols is for such an application. Extensibility, appropriateness, and performance are measures to be evaluated in the next section.

V. Evaluation

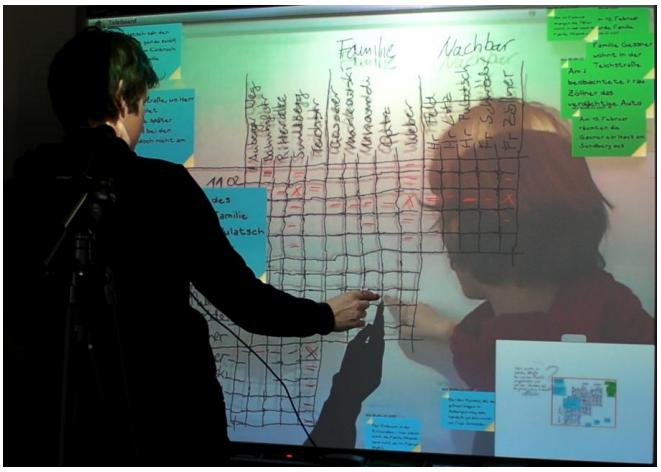


Fig. 5. Tele-Board remote setup including video conference, one participant on the local side, one participant on the remote side

In different user studies (see figure 5) we found out that performance plays an important role in user interaction. People are used to an almost-infinite performance experience with their traditional whiteboards and pens. In earlier project phases we used JavaFX to prototype a whiteboard client interface, which lacked performance at some point. A redevelopment using another UI technology solved that problem.

A more interesting question is, how does the server infrastructure behave in high-load scenarios? How does the addition of a server-wide capturing and analysis plugin influence performance? In the following, we are showing with a series of controlled measurements, how the addition of asynchronous

features influences performance depending on the workload situation of the system.

A. General test setup

The test suite used of the following experiments is a small extension for the whiteboard client. We developed a command line interface which enables us to set the following parameters:

- whiteboard sessions to join in parallel
- whiteboard client threads concurrently in one session
- number of whiteboard change events per second

The purpose of the command line client is to load the whiteboard content from the server and then periodically update the content on the board. This is done to simulate human behavior. We chose 1 update per second as an change interval for all experiments. This is beyond the average workload in typical human-generated sessions, so that the absolute number of concurrently connected clients will be higher in real-world experiments.

The measurements are done directly on the server machine. This server is a Debian Linux based virtual machine limited to 1 GB of RAM - not a high-end device. The observed variables are the CPU load of the MySQL and Openfire processes on the server as well as the transfer rate from and to the server. The measurements are recorded using Unix/Linux built-in tools and averaged over a period of 2 minutes each.

As the whiteboard content used for these experiments we use a blank whiteboard with no drawings on it. Earlier observations revealed that drawings are causing only a little portion of the system load. We placed a set of 25 post-its on the board, each of them with extensive drawings on them. The clients choose randomly which sticky note to change and send the change message to the server, which then distributes it to every other client that update their whiteboard element information.

The following two scenarios will point out two major insights. First, it will be shown which limitations are in a communication model using only one session and connecting as many clients as possible to it and keep it in sync. This scenario is presented to reveal boundaries of the network infrastructure. The second scenario aims more at real-world usage and evaluates the scalability of the system serving multiple parallel sessions each with a constant amount of participants. The idea of both scenarios is to see the influence of the asynchronous in-situ capturing approach on the overall performance.

B. Scenario 1 - many clients in one session

Looking at the concrete scenario 1 of multiple clients in one whiteboard session, it becomes clear that there is potentially a large number of packets to be examined in order to be archived. Figure 6 shows how the message distribution is achieved by the XMPP server. One can see that communication between clients and server is still the major duty of the server infrastructure, even though there is a new server-side plugin being responsible for asynchronous features.

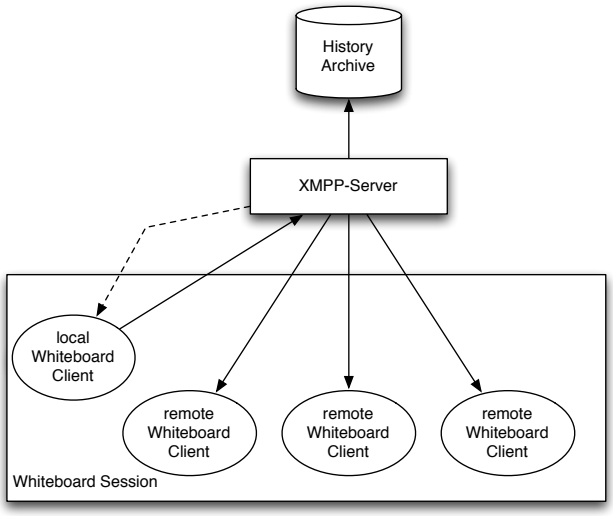


Fig. 6. Conceptual model of a group chat session used for distributed whiteboard interaction

Our evaluation now aims at showing that the server capacity is not limited by the load of the capturing feature. It is more important to look at the transfer rate plotted against the number of clients in one session. In figure 7 right you can see that there is a bottleneck in the number of clients when reaching a limit of about 10MB/s. About 40 to 50 clients seems to be the maximum of concurrent clients in one session. This is due to a 100MBit connection of the virtual machine server used in this setup. The predominant number of messages send in our system (in real-world use cases) are move messages, which are only a few bytes in size. This is also one of the reasons that real-world system load would allow more clients than this experiment.

This high bandwidth requirements are caused by the high number of active participants in the group chat. Thinking of presentation scenarios with few persons being active and many persons only receiving, the numbers would lower dramatically. An example: We have 50 clients producing 1 update per second and the server synchronizes each operation to 49 other clients, we have about $50 * 49 = 2450$ packets to send per second. $10MB/2450 \approx 4kB$ results as an average packet size, which matches the sticky note description size including a path string for the drawing on it. With the same bandwidth in a presentation scenario, theoretically about 2500 receiving clients could be handled with one presenter (1 update$/$s, 2500 receivers, and packet size of $4k$ lead to $10MB/s$).

Nevertheless, the tendency shown in scenario 1 is clear: The limiting factor is not the CPU load of the database for storing the archived communication channel, but it is more the transfer rate from the server, when change requests are propagated to the connected whiteboard clients. The hypothesis, the database operations for enabling asynchronous features really effects the system, cannot hold true in this experiment. MySQL uses a significant amount of CPU time, but the exponential growth

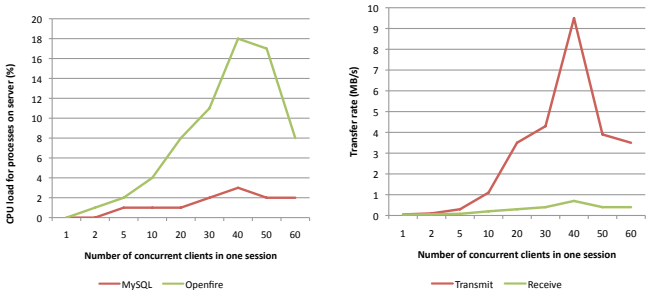


Fig. 7. Scenario 1 - many clients in one session; left: Mysql and Openfire CPU load in relation to the number of clients in one channel, right: the data transfer rate on the server differentiating transmit and receive direction

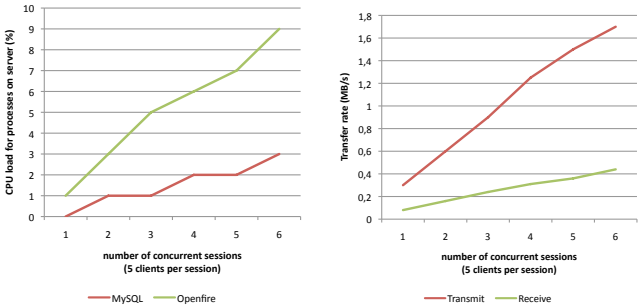


Fig. 9. Scenario 2 - many sessions with equal load; left: MySQL and Openfire CPU load in relation to the number of concurrent sessions each with 5 participating clients, right: data transfer rate on the server vs. the number of concurrent sessions

in transfer rate from the server is much more effecting system performance.

C. Scenario 2 - many session with equal load

This scenario is more realistic for the typical workload on our system. Figure 8 shows that there are multiple sessions running in parallel and few users are logged into the system. For this experiment we chose an equal number of 5 participants in one session. Even for this low number, typical real-world applications will stay under this value, because they often use only two locations (which means two whiteboard clients) at the same time.

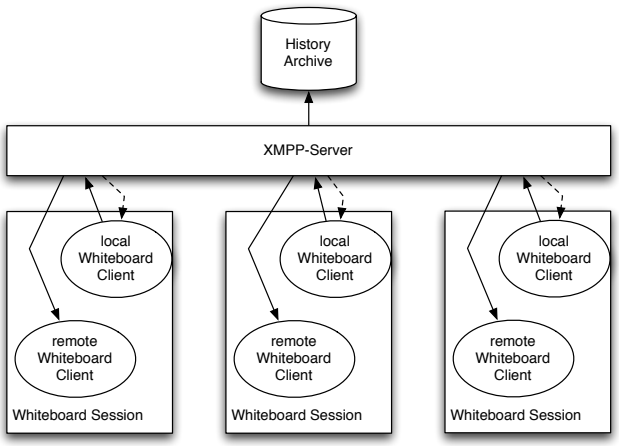


Fig. 8. Conceptual model of multiple group chat sessions used for distributed whiteboard interaction

Figure 9 shows an almost linear development of bandwidth usage and cpu load, the more clients connect. The absolute values - especially for the bandwidth - stay much lower depending on the number of clients (number of concurrent sessions $*$ 5 clients per session) compared to scenario 1. You can see that the system scales very well for this kind of workload. One can also see that the database load is also a fraction of the XMPP server load.

Therefore we can state that the implementation of asynchronous aspects directly on the server does influence all-over performance only little. It adds a proportionate load on the system, but the predominant computation time still is used for message routing to the connected clients. As we have shown in [16], the simple data storing architecture is very appropriate for the typical workload on the system: There are many small write operations and only little longer running read operations. A long and de-normalized table that is used here, optimally supports this setting as it also can be seen in the performance evaluation results.

VI. Conclusion and Outlook

We presented and evaluated the concept and the implementation of an instant messaging protocol that is used for distributed digital whiteboards. Many other tools in this sector focus on one working mode of the CSCW matrix. We try to pursue the goal to enable all modes of interaction equally, so that users can work together co-located as well as distributed and also synchronous as well as asynchronous. The additional workload to be dealt with by the system has shown to be sufficiently low.

The method of capturing and storing complete communication channel traffic in order to rebuild the session afterwards turned out to be highly beneficial for two reasons: It is an efficient way to rebuild what has happened, including the reconstruction of every point in time of the whiteboard history as well as the ability to branch into new parallel sessions. The second advantage is that analyses of the communication scenario - for example for researching on design teams - are much easier to achieve, because nothing related to the communication itself has to be recorded actively, but it is stored implicitly.

Limitations that are shown in the evaluation section are not distracting the usual usage scenarios with our system. Typical setups, when people work together in smaller groups of only a few participants, are well-performing. The assumption of having a large number of participants in one room is strongly hypothetical for our kind of usage scenarios. Even in large presentation scenarios, only few active clients would exist and a potentially large number of receivers. Typical whiteboard

usage scenarios tend not to involve larger numbers than 5-10 participants.

Future applications based on the archive we develop over time aim in the direct of better analyzing the gathered communication data. For example it is unnecessary for a person who wants to understand what has happened in past sessions, to watch at every single event. Possible analyses and visualizations are based on the assumption that there are states which are more important than others and should be highlighted. Based on observation and user feedback we have to come to a common understanding of an important state.

The generalization of this method can be an approach for other applications as well and is not limited to whiteboard use. Overall, XMPP as an instant messaging protocol turned out to be well-suited for synchronizing digital whiteboards and the archived communication data carries a large potential for further research. The results of our evaluations play an important role also for gathering insights on deployment of the Tele-Board suite into business environments with high demands on reliability.

ACKNOWLEDGMENT

The authors would like to thank the support of the HPI-Stanford Design Thinking Research Program. I would especially like to thank Matthias Quasthoff, Raja Gumienny, and Markus Dreseler.

REFERENCES

[1] T. Brown, "Design thinking," *Harvard Business Review*, June 2008.

[2] R. Johansen, *GroupWare: Computer Support for Business Teams Export GroupWare: Computer Support for Business Teams*. The Free Press, 1988.

[3] C. A. Ellis, S. J. Gibbs, and G. Rein, "Groupware: some issues and experiences," *Communications of the ACM*, vol. 34, no. 1, 1991.

[4] T. Rodden, "A survey of CSCW systems," *Interacting with Computers*, vol. 3, no. 3, pp. 319–353, 1991.

[5] H. Ishii and M. Kobayashi, "Clearboard: A seamless medium for shared drawing and conversation with eye contact," *CHI*, 1992.

[6] J. C. Tang and S. Minneman, "VideoWhiteboard: video shadows to support remote collaboration," in *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*. ACM New York, NY, USA, 1991, pp. 315–322.

[7] K. M. Everitt, S. R. Klemmer, R. Lee, and J. A. Landay, "Two worlds apart: bridging the gap between physical and virtual media for distributed design collaboration," in *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2003, pp. 553–560.

[8] Coccinella — instant messaging program with whiteboard - http://thecoccinella.org/.

[9] M. Bengtsson. Memo: Svg & xmpp - http://coccinella.sourceforge.net/docs/memosvg_xmpp.txt.

[10] Jep-xxxx: An svg based whiteboard format, http://xmpp.org/extensions/inbox/whiteboard.html.

[11] M. Parparita and S. Rusinkiewicz, "Thor: Efficient whiteboard capture and indexing," Princeton University, Tech. Rep., 2004.

[12] (2004) Jabber software foundation network working group - extensible messaging and presence protocol (xmpp): Instant messaging and presence.

[13] Google, "Google talk. http://www.google.com/talk/."

[14] Psi, "The cross-platform jabber/xmpp client for power users. http://psi-im.org/."

[15] W. Geyer, J. Vogel, L.-T. Cheng, and M. Muller, "Supporting activity-centric collaboration through peer-to-peer shared objects," *GROUP*, pp. 115 – 124, 2003.

[16] L. Gericke, R. Gumienny, and C. Meinel, "Message Capturing as a Paradigm for Asynchronous Digital Whiteboard Interaction," in *6th International ICST Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2010.