MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Instant Co-Browsing Lightweight Real-Time Collaborative Web Browsing

Alan W. Esenther

TR2002-19 May 2002

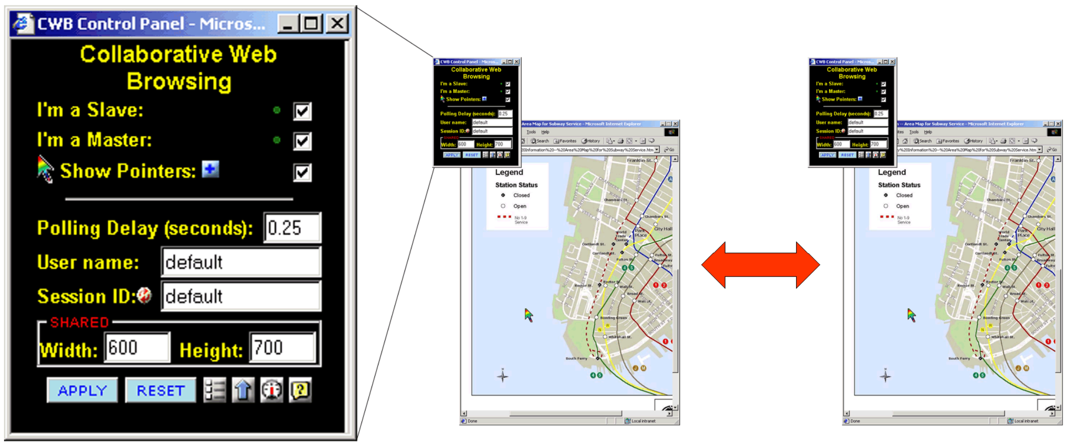
Abstract

A lightweight collaborative web browsing system, that targets casual (non-technical) web users, is presented. This system allows remote participants to easily synchronize pointing, scrolling and browsing of uploaded content in their web browsers. Since instant messenging systems have become a very popular method for remote participants to engage in real-time text chat sessions, it is conjectured that this simple co-browsing system which allows remote participants to share and point at their pictures and web content (instead of just sharing text) could prove useful and fun, too. The collaboratively viewed web content could either pre-exist on a host web server or, in a more typical scenario, be dynamically uploaded by the remote participants themselves. A specific goal of this system is to keep the interactions extremely simple and safe. Any user should be able to use it intuitively with a single click; there are no pre-installation or pre-registration requirements. This system is based on simple polling-based scripting techniques that avoid intrusive mechanisms based on proxies or events. Most significantly, there is no reliance on any controls, applets, plug-ins or binary executables, since these would require the trust of participants and are virus-prone. It is the reliance upon such inconvenient helper programs, along with any pre-installation or pre-registration requirements, that makes existing co-browsing offerings more "heavyweight", and limits their appeal for casual collaboration.

WWW2002 Conference Proceedings

Instant Co-Browsing: Lightweight Real-Time Collaborative Web Browsing

Alan W. Esenther
Mitsubishi Electric Research Laboratories
201 Broadway
Cambridge, MA 02139 USA
esenther@merl.com



I'm a Slave: ✓
I'm a Master: ✓
Show Pointers: ✓
Polling Delay (seconds): 0.25
User name: default
Session ID: default
SHARED
Width: 600 Height: 700

Figure 1: A representation of a CWB collaborative web browsing session. On the left is a close-up of the CWB Control Panel. This is a pop-under window that users never actually have to see. As any user in the session browses, scrolls, points, or fills in a form in any shared collaboration window (the windows displaying a map image above), the changes occur in everyone else's browser, too.

ABSTRACT

A lightweight collaborative web browsing system, that targets casual (non-technical) web users, is presented. This system allows remote participants to easily synchronize pointing, scrolling and browsing of uploaded content in their web browsers. Since instant messenging systems have become a very popular method for remote participants to engage in real-time text chat sessions, it is conjectured that this simple co-browsing system which allows remote participants to share and point at their pictures and web content (instead of just sharing text) could prove useful and fun, too. The collaboratively viewed web content could either pre-exist on a host web server or, in a more typical scenario, be dynamically uploaded by the remote participants themselves. A specific goal of this system is to keep the interactions extremely simple and safe. Any user should be able to use it intuitively with a single click; there are no pre-installation or pre-registration requirements. This system is based on simple polling-based scripting techniques that avoid intrusive mechanisms based on proxies or events. Most significantly, there is no reliance on any controls, applets, plug-ins or binary executables, since these would require the trust of participants and are virus-prone. It is the reliance upon such inconvenient helper programs, along with any pre-installation or pre-registration requirements, that makes existing co-browsing offerings more "heavyweight", and limits their appeal for casual collaboration.

Keywords

Web collaboration, synchronized browsing, real-time distributed collaboration, shared web browsing, instant messenging, co-browsing

Word count

Approximately 7500 words

1. INTRODUCTION

Real-time web collaboration software solves the problem of how to allow multiple users to synchronize their views of web pages. For example, if any of the users in a collaborative session browses to a new page, scrolls a page, drags a shared pointer, or types in a form input field on a page, then all of the other users will simultaneously see that change in their browser windows, too (see Figure 1).

Typically the users would be talking on the telephone while collaborating, although some form of Internet telephony might be used instead. This paper presents a lightweight approach to collaborative web browsing, herein referred to as CWB (Collaborative Web Browsing). CWB does not require any virus-prone binary programs, applets, controls, or browser plug-ins. It is based on a polling architecture and Dynamic HTML (HTML, CSS and Javascript) [1]. It is intended to facilitate spontaneous collaboration between arbitrary users, from arbitrary locations on the Internet, using web browsers on arbitrary computers. As such, it can be applied as a means for "instant co-browsing" - allowing users to collaboratively browse instantly. As with instant messenging systems (such as AOL Instant Messenger [2], ICQ [3], MSN Messenger [4], and Yahoo! Messenger [5]), CWB is intended for "casual" collaboration -- safe, immediate sharing of user content between arbitrary users (perhaps strangers). In some sense CWB is more "instant" than instant messenging applications because no pre-installation or helper program download is necessary to co-browse via CWB. Note that one requirement incurred by the exclusive use of Javascript (without any applets or controls) is that only content that resides on the same web server as the CWB scripts can be co-browsed. CWB is not designed for co-browsing to arbitrary websites. Rather, inspired by instant messenging systems, CWB is designed to facilitate spontaneous interaction with content that is dynamically uploaded by session participants. In some sense this is an effort to push the limits of the co-browsing experience that can be provided simply by fetching a plain HTML web page. The user is never prompted to grant permissions to an ActiveX control, a Java applet, or an installer for a binary executable -- yet the user can co-browse immediately. Also note that this approach is intended for sharing web page content and should not be confused with distributed whiteboard applications that allow sharing of scribbles, text, and pictures only on a special (non-HTML-based) electronic canvas. This paper presents CWB along four dimensions: collaboration features, user interface, page non-interference mechanism, and security. The design philosophy for each of these dimensions can be briefly summarized as follows:

Collaboration Features:
"1-click collaboration" that replicates every detail of the shared web page, and adds shared pointer support.
User Interface:
The best UI is no UI -- a simple pop-under control panel is available, but no user should have to see or interact with it.
Page non-interference mechanism:
Only read - don't modify -- the shared web pages.
Security:
Don't use any (virus-prone) helper programs (e.g. controls, applets, plug-ins, etc.).

2. EXISTING SOLUTIONS

Existing web collaboration solutions may be broadly classified in terms of how "lightweight" they are. A more heavyweight solution has more demanding requirements that must be met before collaboration may begin (e.g. software pre-installation, user pre-registration).

Heavyweight solutions such as Microsoft NetMeeting [6] rely on operating system support to recreate the user's entire screen (or just one or more open windows) across the network. This insures that all users are seeing precisely the same view of the web browser (or of any application). However, while appropriate in many cases, this solution may not be ideal when the goal is spontaneous casual collaboration from anywhere. NetMeeting is a proprietary solution that only runs on the Windows platform. It requires each user to pre-install a binary application on their computer, and to pre-register before collaboration may begin.

Another class of heavyweight solutions, such as the Albatross system [7] and GroupWeb [8] embed synchronization mechanisms directly into the WWW client. This approach, however, obviously adds the requirement that each user use the special WWW client. Rather than requiring a whole new WWW client, it is also possible to add a plug-in to an existing standard browser to help coordinate synchronizations. This was the approach used for a web collaboration home banking system [9]. A plug-in can also be considered heavyweight, though, since it requires pre-installation and browser modification. A plug-in also increases the disk and memory footprint of the browser just so that it will work with sites designed to use that particular plug-in. Certainly there are cases, though, where these heavyweight approaches are acceptable and suitable.

Even a lighter-weight solution, such as that provided by Hipbone [10], downloads a Java applet into the user's web browser. A web collaboration banking kiosk system (also described in [9]) and the Artefact framework [11] are additional examples of applet-based solutions. This approach obviously requires Java support on the client computer, and it adds delays while the Java environment is loaded. Also note that the Hipbone product, as well as a similar offering by WebDialogs [12], is targeted at customer support centers. As is typical for this type of product, the collaborative session is coordinated via a binary executable client program that is installed and run on a customer service representative's computer. This may not be suitable for casual users who wish to collaborate quickly and independently, without the requirement that one of them runs a coordination program. In this respect, collaboration via CWB is perhaps analogous to instant messaging: if two instant messenger users wish to exchange text messages, there is no need for a third party (a customer service representative) to be present in order for the session to proceed.

Another common requirement that is prevalent with existing solutions is that users have to take turns making changes. Typically the user interface has some type of control to specify which user is currently allowed to make changes (such as browsing to a new page or scrolling the page). A more flexible solution would allow any user to make a change at any time, rather than first having to request the privilege. If the users have to pass some token to allow them to make changes, then their attention must be diverted from the shared web page to the token control, which distracts them from the collaborative session.

An even lighter-weight solution (used by CWB) could use just Dynamic HTML scripts to peek into the shared web page and extract and apply changes. A patent by IBM [13] explores this idea, although it relies on a Java applet in the client to coordinate updates between the client web browser and the web server. That solution may be characterized as "intrusive", however, because it modifies the contents of the shared web page by placing it into a special layer. This is done so that layers with pointers and other collaboration elements can be placed above the shared web page. However, this technique can break the behavior of scripts in the target web page. Such scripts might not expect that the containing document has moved to a new layer.

An older mechanism, also mentioned in the IBM patent, uses a proxy approach. Before a web page is sent to the client, a web server-based proxy program rewrites all of the URLs in the page such that they reference the collaboration web server. The CoWeb system [14] also uses a proxy approach, though it replaces HTML elements with Java applets. However, this is even more intrusive and more likely to change the behavior of the existing web page, partly because it is no trivial task to detect all of the myriad ways in which the HTML and scripts in an arbitrary shared web page might create external references. Pages that use scripts to dynamically create new external references and HTML elements are particularly problematic for these systems.

Another technique for monitoring changes within a shared web page is to use event handlers. A script inserts or chains it's own event handler into various elements of the web page, essentially creating a callback to the monitoring script. (The WebVCR system [15] uses this technique to send information about user actions to a companion applet rather than a script.) As an example, the monitoring script might want to intercept scroll events so that it will know when to replicate scrollbar changes to other users. This technique can also be considered intrusive and may change or break the behavior of the web page. For example, the callback may execute in response to an event before returning control to the shared page's event handler. However, the shared page's handler might abort the event, or trigger some change that might not be detected and thus might not be replicated to other users. A script in a shared page might also dynamically create a new page element that the monitoring script will not know about.

Finally, existing solutions may not support all of the fine-grained collaboration features discussed later.

3. HIGH-LEVEL ARCHITECTURE

Most of the novelty of the CWB solution is embodied in the design philosophy used for each of four dimensions of web collaboration, as is described in subsequent sections. The high-level architecture is described in this section.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.