TRACKING AND PRESENTING USER ATTENTION FOR COLLABORATIVE BROWSING USING HETEROGENEOUS DEVICES

Michelle Yan†, James Shaw†, Vahid Khamsi‡, and Shih-Ping Liou†

1 Siemens Corporate Research, Inc.
755 College Road East
Princeton, NJ 08540 USA
{myan,shaw,liou}@scr.siemens.com

2 Eurecom Institute
Multimedia Information Processing
Sophia-Antipolis, France
Vahid.Khamsi@epfl.ch

ABSTRACT

This paper presents mSpiders, a system that extends traditional co-browsing to support mobility and awareness among participants. The system offers mobile users the ability to easily get help and share web documents with others over the Internet and greatly enhances users’ web-sharing experience. By introducing a new method to track and present the current attention of each participant – their region of interest (ROI), the system enables all participants aware of each other’s presence and what they are viewing, independent of the specificities of Web content, browsers used and their settings.

1. INTRODUCTION

A co-browsing system is a communication and collaboration tool that enables a user to get help or inform other participants through automatic synchronization of the Web pages being viewed. Many co-browsing systems on the market are primarily used as a Customer Relationship Management (CRM) tool – where a representative waiting at the other side to guide a customer to the information being sought over the Internet. For example in UgoCo product (www.ugoco.com), the tour guide can take control and automatically lead tour participants from URL to URL using Co-Browser Guide. However, because these systems concentrate on desktop sharing and ignore mobile users, the potentials of the systems are greatly limited. Since in the current co-browsing systems, each agent must be in the same physical location, attracting users to use such a co-browsing system again and again.

The paper is organized as follows. We first review related work in co-browsing for desktops and for heterogeneous devices. Next, we provide an overview of the mSpiders system. We then describe the details of our approach to track and present ROI, and related results for heterogeneous devices. Finally the paper ends with a conclusion.
2. RELATED WORK

Because of strong industry interest, detailed descriptions of co-browsing systems are hard to come by in publications [2,3,4,5]. Many co-browsing systems use plug-ins or native code requiring software installation or specific security permissions to be granted before customers can use them. Since such an approach is browser and platform dependent, it hinders customer's effort to use the co-browsing system. Instead, we use proxy server to add co-browsing capability in the Web environment, as proposed in [4,5]. This paper extends such architecture to incorporate features that improve user's experience while co-browsing using heterogeneous devices.

Previous works addressed many issues of tailoring information presentations in a collaboration environment with heterogeneous devices. For example, R. Han et al. [6] focused on personalized partial views for heterogeneous devices depending on the access privileges of the user. The framework of Marsic et al. [7] transforms information to match client’s local capabilities and resources, yet maintaining important content information in a 3D to 2D collaborating environment. In [8], Rist et al. use automated presentation planning for the generation of views in a collaboration environment with heterogeneous devices. Different from these different works, we assume that recent growth in network bandwidth and device capabilities will allow mobile devices to download full HTML pages. As a result, instead of tailoring presentations to generate different views for different devices, our focus is on providing awareness through tracking users’ current attention and informing others in a synchronized fashion during a co-browsing session using heterogeneous devices. We utilize features provided in the Document Object Model (DOM) [9] of HTML page and track mouse or stylus events of each individual device.

3. OUR CO-BROWSING INFRASTRUCTURE

*mSpiders* is a proxy-based co-browsing system that enables users to share personalized web pages without installing client software. As shown in Figure 1, the co-browsing server sits before an original Web server and intercepts all the HTTP requests from users’ browsers. We divide co-browsing server into two major components – a traditional co-browsing component and a ROI component. We first briefly describe the traditional co-browsing component designed for desktop sharing. The ROI component, which focuses on co-browsing for heterogeneous devices, is described later in this section.

To enable synchronization among the browsers, *mSpiders* inserts JavaScript and a Java hidden applet into all HTML pages downloaded from the original Web server before they are forwarded to a user’s browser. The hidden applet performs the logics that turn customer’s browsers into active participants synchronizing the content displayed in their browsers in co-browsing sessions. In this arrangement, owner of a website adds co-browsing functionalities to his web site without manual modification to any document on the web site. Users still browse the website as usual, but they have extra co-browsing functionality to share the content without downloading any plug-in, installing software, or changing any setting in his browser.

**Figure 1:** *mSpiders* supports co-browsing by intercepting HTTP messages between original Web server and browsers on heterogeneous devices.

Inside *mSpider* system, the traditional co-browsing component consists of three modules: Session Manager, Cache Manager, and Cookie Manager. The main task of the Session Manager is to keep track of active co-browsing sessions and decides on the appropriate actions to take to fulfill customers' HTTP requests. For example, when a HTTP request is received, Session Manager identifies which co-browsing session the request belongs to and decides if it should get the content from the cache, or retrieve it from the original server through the Cache Manager. The Cache Manager retrieves both static and dynamic documents from the original server and performs session-based caching – a separate cache is created for each co-browsing session. This ensures other participants see the exact same page when joining a co-browsing session. Cookie Manager manipulates cookies from both requests from client browsers and replies from the original web server to ensure proper cookie information is transmitted between the browsers and the original server.

**Figure 2:** Details of ROI component and its interactions with other components.

The ROI component, as shown in Figure 2, is specifically designed for handling the issues in mobile co-browsing, and it consists of three modules: Java Applet Inserter, ROI Manager
and ROI Database. The Java Applet Inserter is responsible for inserting JavaScript and the Java hidden applet into all HTML pages from the original web server. In addition to performing the tasks described previously, the hidden applet also tracks mouse or stylus events of each user, and detects his current attention – ROI and transmits such information to the server. The ROI component interacts with the traditional co-browsing component to monitor relevant status related to ROI. The ROI Manager also communicates directly with the hidden applet in the client browser for real-time tracking and updating what is the focus of attention of each user using HTTP messages. Once received a request from the applet, the ROI Manager immediately goes to the ROI Database, and makes the corresponding changes to the table associated with the particular user if necessary, and checks any changes made by other participants, and transmits back such changes to the hidden applet in the client for update.

4. CO-BROWSING USING HETEROGENOUS DEVICES

In heterogeneous device co-browsing environment, difference in screen size and in user interactions with devices present interesting challenges. For example, a desktop user can view major portion of a Web page at a time, but a PDA user may have to keep scrolling the page to locate the information being sought. It is not straightforward to provide the same context for all participants to support collaborative browsing for heterogeneous devices. To provide mutual context, we introduce a way to track and identify user’s current attention that can be communicated to others in the same co-browsing session. We denote a user current attention as the region of interest (ROI), which can be the paragraph they are reading, the image they are interested in, or the object they are interacting with.

4.1 REGION OF INTEREST

As discussed in Section 1, how an HTML page is displayed can be very different because of different devices capabilities or browser settings. So to support mobile co-browsing, one must find and capture unique features that come from an HTML page common to all devices and represent the basic elements of a HTML page – paragraphs or images.

Document Object Model (DOM) elements - the basic data elements of a HTML page offer the properties that satisfy our requirement, and are supported by all major Web browsers. Therefore, we define a DOM element as the unit of ROI, as shown in Figure 3. A particular ROI is identified at the client side by the hidden Java applet, based on the combination of features provided in the DOM of HTML page and a mouse or stylus event of a participant. It is independent of devices, web content, browsers used and their settings. For example, in the desktop case, once a mouse is over a DOM element in a HTML page longer than a certain time threshold, the hidden applet in browsers detects such a event and then automatically highlights the element in the HTML page. At the same time, the applet transmits the element’s ID to the server as a ROI of this particular participant. The server monitors changes and sends the updated element ID to other participants. At the reception of a new ROI, the displays of the other participants are automatically scrolled to bring to focus of the ROI, and the corresponding element is also automatically highlighted using a different color.

4.2 SCANNING TECHNIQUE FOR MOBILE DEVICES

Figure 3: An original HTML page and the corresponding DOM elements – the potential ROIs.

Figure 4: the mesh points used for detecting ROI of mobile device and the identified ROI on a mobile device and desktop browser.

A user interacts with a mobile device using a stylus and screen tapping, and the display screen is very limited. Often the case is that a small portion of a paragraph is visible, users then have to scroll vertically and horizontally to see the rest of the paragraph. So for mobile devices, we define the ROI as elements of an HTML page that are visible in the display screen. However, Web browsers do not allow us to search what DOM elements are in a given display window or region; we can only search if there is a DOM element at a given location in
display screen. As a result, we cannot infer directly which DOM elements are visible in the display screen.

To identify a user’s current attention, we develop a scanning technique to compute all visible elements of a HTML page on a mobile device. The algorithm can be described as follows. Once a HTML page is not scrolled for more than a certain time interval, the display screen is automatically scanned using a mesh of points over the active display area, see Figure 4. A collection of distributed points is overlaid onto the display area, so the DOM elements can be identified at these given mesh points on the window. The union of all detected DOM elements is defined as the visible elements – ROI. In our implementation, a mesh of points is built and distributed linearly along the horizontal and vertical direction with a distance of 30 pixels between each pair (about 50-80 points for a PDA sized screen). DOM elements identified at these locations on the window are used to compute the ROI. Same as in a desktop case, the elements are then automatically highlighted and their IDs are transmitted to the server to inform other participants for update.

4.3 PRESENTING A REGION OF INTEREST

Figure 5: Visual effect associated with each of the three stages – idle, browsing and the current attention.

Figure 5 is a screen capture of mSpiders to demonstrate the process of tracking, gathering and presenting the region of interest. In this process, the hidden applet in the browser constantly monitors the mouse or stylus activity, and captures the activity, processes it and deduces the ROI. A user in a co-browsing session can be in one of the three states -- idle, browsing, and ROI as shown in Figure 5. In the idle state, the mouse or stylus is out of the HTML page display screen or not passing over any DOM element. In the browsing state, the mouse or stylus is quickly passing over a DOM element, and the element is then highlighted in beige color to indicate the browsing state. The system goes into ROI state when a mouse or stylus stays on a DOM element or a combination of several DOM elements for more than a certain time interval. It is assumed that this element is the user current attention – ROI. The detected element(s) in a HTML page - ROI is automatically highlighted with a darker color to give the user visual feedback. At the same time, the ROI manager on the server side is notified of the change, and informs hidden applets in other browsers to make the corresponding changes.

5. CONCLUSION

mSpiders presented in the paper extends the potential of traditional co-browsing by supporting mobility and awareness among users. We achieve this through tracking and presenting each user’s attention – ROI using mesh points and DOM. By providing both mobility and awareness, mSpiders offers a convenient tool to attract more business from entertainment, consumer as well as enterprise.

7. REFERENCES