

Synchronized Web Surfing:

A Web-Based Application for Interaction between Online Users

Tao Sun, Yang Li, Bill Tucker
*Broadband Applications and Networks Laboratory, Department of Computer Science
Faculty of Science
University of the Western Cape (UWC)
Private Bag X17 Bellville 7535 South Africa
E-mail:{2336532,2347240,btucker}@uwc.ac.za*

ABSTRACT

Synchronized Web Surfing (SWS) is a new web-based application. It enables two or more users to keep on sharing the content through their automatically synchronized web browsers. As an independent component, SWS can be easily integrated into other network applications, such as Call Center, Web Conference and Distance Learning. This paper presents the architecture and the requirements that the application has to fulfill. Using Call Center application with SWS as an example the service flow is described. Finally, future work is planned.

KEYWORDS

Servlet, Applet, HTTP, Session, E-Commerce, Synchronize

1. INTRODUCTION

Web-based applications have changed traditional communications mode dramatically, typical examples of which are Email, E-Commerce [5], Distance Learning [6] and Web Conference. However, the demand for more new web-based applications is growing for online communication. In some case, during Web Conference or Distance Learning, participates would share their WebPages or documents with others synchronously. In another case, while visiting company's website or doing online shopping, users are often confused by multifarious information that they can not find out exactly what they want in good time. This is where the concept of Synchronized Web Surfing (SWS) comes in.

SWS allows two or more users anywhere on the Internet to join in a so-called SWS session, where the desired content is displayed by their automatically synchronized Web browsers. Owe to its independency, SWS system can be easily integrated into other applications (e.g. Web Conference or Call Center [7]), which results in the improvement of its online interaction capability with other applications.

SWS system work as a web-based application by adopting Java Applet and Servlet technologies. Applet [1] is a small Java program that can be embedded in an HTML page. Its common rule is to make an Internet connection to the computer from which the Applet was sent. Servlets [3] are modules of Java code running in server application (hence the name "Servlets" on the server side is similar to "Applets" on the client side). Their task is to answer client requests. Servlets are not tied to some specific client-server protocol though normally they are used with HTTP. The word "Servlet" is often explained as "HTTP Servlet".

2. ARCHITECTURE

SWS system is based on Client/Server architecture [2]. The client is an Applet packed in the web page, which communicates with Servlets of SWS Server through HTTP interface [4]. The following figure describes the functional architecture of SWS system.

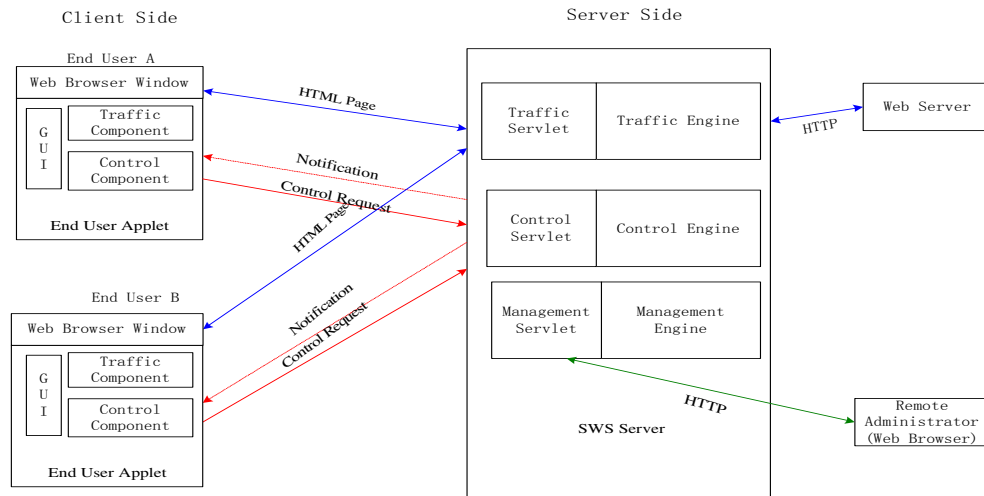


Figure 2.1: Functional architecture of SWS system

2.1 Server side

2.1.1 Control component

SWS control component manages the SWS sessions and controls each session. It includes two parts:

- **Session Control Engine:** It controls and manages all SWS sessions, including creating, joining, leaving, deleting a session, granting control, ungranting control, taking over a session and getting session status. For each SWS session, a timeout is set to end session. If the SWS Control Engine doesn't receive any request from end users during the period of timeout, the session shall be cancelled.
- **Session Control Servlet:** Servlet communicates with the end user Applet. After having received session control HTTP requests from end user Applet, this Servlet calls the appropriate functions of Session Control Engine, and then transmits Session Control Engine responses to the client Applet over HTTP.

2.1.2 Traffic component

SWS traffic component is designed to manage HTML traffic processing. It is composed of Traffic Engine and Traffic Servlet.

- **Traffic Engine** manages to synchronize the content of the listener's web browser with the one of the speaker (User who controls the browsing at a given time is called the "Speaker"; all other users who are in the same session are called the "Listeners"). The synchronization process is as following:
 - receives the speaker's HTTP requests,
 - forwards the requests to web site and stores the HTTP response,

- *changes* the HTML links and forwards the response to the speaker,
 - *parses* the received HTML page to disable the links and stores the page,
 - *indicates* the control engine to send a "new URL" notification to the listener Applets,
 - *supplies* the stored page (with inactive links) to each listener.
- Traffic Servlet receives the client HTTP requests, then calls the appropriate functions of Traffic Engine and transmits Traffic Engine responses to the client over HTTP.

2.1.3 Management component

SWS Management module is in charge of SWS server management. It comprises Management Engine and Management Servlet.

- Management Engine: offers functions to manage the SWS server. They include:
 - *List* running sessions characteristics,
 - *Allow killing* a specific session,
 - *Allow sending* a text message to the SWS end users,
 - *Generate* operator logbook and session tickets.
- Management Servlet: this Servlet manages the remote operator data that are stored in the configuration file. The Management Servlet receives the management HTTP requests from the remote manager, calls the appropriated functions of Management Engine, modifies eventually the configuration file parameter value(s) and transmits Management Engine responses to the remote manager over HTTP.

2.2 Client Side

To communicate with the SWS server, an Applet is downloaded to client side through an HTML page. The client Applet includes three parts:

- Graphical End User Interface (EUGUI): It communicates with End User Control Component and End User Traffic Component to deal with end user's inputs and to provide him the session status.
- End User Control Component (EUCC): It allows the end user to control some aspects of the SWS session and display current session status. Control functions include taking over a session and leaving a session. When it receives a "new state" notification (e.g. a new user) from SWS server, it sends a "status change" event to the "EUGUI". The "EUGUI" then refreshes these corresponding fields in terms of new status value. When it receives a "new URL" notification, it transmits this URL to the "EUTC".
- End User Traffic Component (EUTC): At the beginning of a SWS session, it opens a browser window. The window just contains the synchronized web pages. If the end user is the speaker, he surfs in this window. If the end user is a listener, he watches the speaker surfing in this window. This component refreshes the browser window according to the end user requests provided by the "EUGUI" (if the end user is a speaker) and URL provided by the "EUCC" component (if the end user is a listener). This component also records the user URL history. Each URL displayed in the browser window is added to the user history. When the session ends, it closes the browser window.

3. SERVICE FLOW EXAMPLE

Using the Call Center application with SWS as an example, the service flow of SWS is described in detail as following. In the scenario that the client is browsing the website of a company, he is interested in or puzzled at some information, he can contact with the call agent by the hotline. Utilizing the SWS service, the call agent can assist the Client.

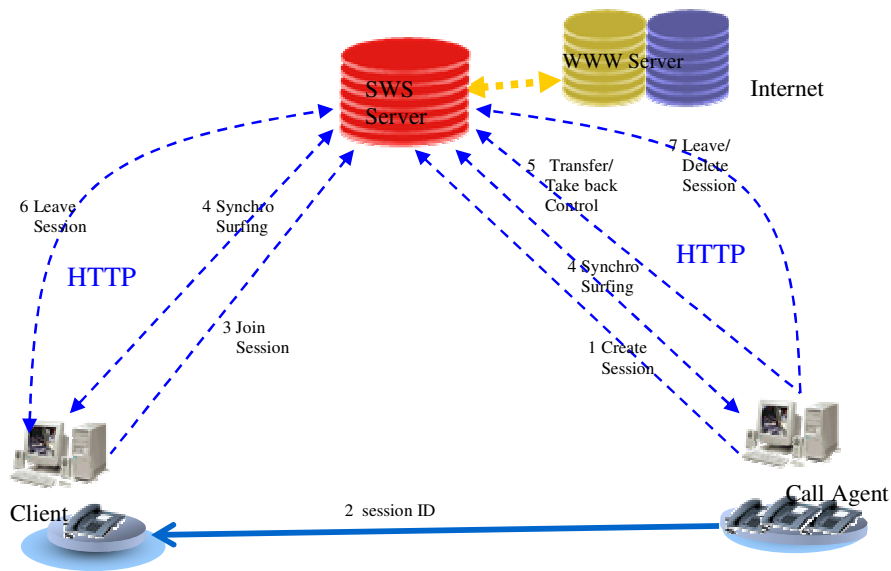


Figure 3.1: Service Flow of SWS system

Step 1, 2: After creating a SWS session through the “Create Session” webpage, the call agent tells the client about the session ID by phone. Meanwhile, a control panel window and the default browsing window will appear and the navigation control is in the call agent side.

Step 3: According to the session ID, the client can join the session through the “Join Session” webpage. Meanwhile, a control panel window and the default browsing window which is the same as call agent will be displayed, but the action buttons on the panel and the hyperlink in the web page are disabled.

Step 4: By inputting website address or clicking “Up”, “Down” button on the control panel, the browsing window will change to the specified website synchronously in both the client and the call agent’s agent side. So that the call agent can help the client to find out what he want. Note that the hyperlink of the webpage in the client side is disabled.

Step 5: The call agent also transfer the control to the client by clicking the “Granting Control” button on the control panel. So the client can do the same actions to navigate the website as what the call agent does in step 3. The call agent also can draw back the navigation control from the client by clicking the “Ungranting Control” button on the control panel.

Step 6, 7: After finishing SWS service, the call agent or the client can exit by clicking the “Exit” button on the control panel. If the client does so, he leaves the SWS session; if the call agent does so, that means he leave the SWS session and delete it at the same time.

4. FUTUER WORK

Project ‘Synchronized Web Surfing’ is currently implementing the basic functions described in this paper. We plan to extend the functions of SWS in the following aspects:

- Improve parsing capability of links, not only for the static links that is contained as <a href> elements in the HTML source code, both also JavaScript based functions dynamically generating the links

- Improve handling capability to dynamic action on the webpage. For instance, insertion of text in forms or selection of options in HTML forms, the change of these HTML objects are not reported back to the SWS server.
- Finding out the specific requirements of different applications with SWS, for example, the coordination to blackboards and workspace in CSCW application.

5. CONCLUSION

Synchronized Web Surfing provides a bran-new online interaction mode. It enables more than one online user to browse the automatically synchronized Web page. Using Java Applet and Servlet technology, the user can use the service without installing any software, just through the web browser. It makes the application easy to use. It can be applied in Call Center, Distance Learning, Web Conference and other Internet Services. These SWS enabled applications make online communications more flexible and intelligent.

REFERENCES

1. Bruce Eckel, President, MindView, Inc. 2000. *Thinking in Java Second Edition*. Published by Prentice Hall PTR, One Lake Street, Upper Saddle River, New Jersey
2. Danny Ayers, Sing Li, Paul Houle, Mark Wilcox, Ron Phillips, Piroz Mohseni, Stefan Zeiger, Hans Bergsten, Matthew Ferris, Jason Diamond, 2001. *Professional Java Server Programming with Servlets, JavaServer Pages (JSP), XML, Enterprise JavaBeans (EJB), JNDI, CORBA, Jini and Javaspaces*. Published by Wrox Press Inc, 10475 Crosspoint Blvd. Indianapolis, IN 46256
3. Andrew Harbourne-Thomas, Sam Dalton, Simon Brown, Bjarki Holm, Tony Loton, Meeraj Kunnumpurath, Subrahmanyam Allamaraju, John Bell, Sing Li, 2002. *Professional Java Servlets 2.3*. Published by Wrox Press Inc, 10475 Crosspoint Blvd. Indianapolis, IN 46256
4. CHENG Shao-fei and WANG Hong-wei, 2002. The Mechanism of the Communications between Applet and Servlet. *In Software Technique, O. I. Automation*, Vol. 21, No. 4
5. E-Commerce Tutorial Available from <http://hotwired.lycos.com/webmonkey/e-business>
6. What is Distance Learning? Available from <http://www.cdiponline.org/dlinfo/cdip1/distance/home.html>
7. The Development of Call Center Technology and Application. Available from http://www.ctiforum.com/forum/2003/05/forum03_0532.htm