Collaborative Web Browsing Tool supporting Audio/Video Interactive Presentations

Guillermo de Jesús Hoyos-Rivera§, Roberta Lima-Gomes+ & Jean-Pierre Courtiat
LAAS-CNRS
7, Av. du Colonel Roche, 31077, Toulouse, France.
{ghoyos|rgomes|courtiat}@laas.fr

Roberto Willrich*
INE-UFSC
C.P. 476, 88040-99, Florianópolis, Brazil
willrich@inf.ufsc.br

Abstract

Collaborative Web Browsing aims at extending currently available Web browsing capabilities in order to allow several users getting their browsing activities synchronized. A Collaborative Web Browsing system should provide all the necessary facilities to allow users to get synchronized and desynchronized in a flexible way, including both browsing actions and browser embedded presentations interactions. In this paper we present the modeling and implementation of our Collaborative Web Browsing system called CoLab 2. It implements the necessary functionalities that allow solving the problem of allowing users to collaboratively browsing the Web.

Keywords: Collaborative, Web, Browsing, Synchronization, Continuous Media.

1. Introduction

Collaborative Web Browsing can be seen as an extension of traditional Web browsing. The latter consists essentially of users accessing resources that are available in Web servers connected to a network (Internet or Intranet). Normally users are isolated when browsing the Web since they have no way of sharing online their browsing activities with other users. Collaborative Web Browsing overcomes this problem allowing users to “browse together”.

In [1], we proposed a Collaborative Web Browsing system called CoLab, which allows users to easily create and release browsing synchronization relations. We understand a synchronization relation as binding the Web browsing of one user to that of another one. The user who gets synchronized is then called synchronous user. In this way the browser of the synchronous user will automatically retrieve and present the same Web pages sequence as the one requested by the user he is synchronized with. Synchronization relations are created and released by using of some predefined synchronization operators.

An important fact that we have taken into account is that the number of Web pages containing continuous media increases quickly. This is the case notably in the e-learning field because the use of continuous media allows to present complex and abstract concepts and helps keeping students attention.

In this paper we present an extension to our original Collaborative Web Browsing system, which includes, besides a browsing synchronization facility, the synchronized presentation of audio and/video media streams embedded in Web pages. This capacity is particularly useful in e-learning where, for instance, the instructor can make use of video-based learning objects embedded in Web pages and control the presentation seen by the students in the same fashion as in traditional classes.

The paper is organized in 6 main sections. In section 2 we present an overview of the Collaborative Web Browsing field and discuss the related work, justifying then our approach. In section 3 we present the architecture of our system and detail the mechanisms to allow users to create and release synchronization relations, as well as to guarantee the synchronized presentation of interactive continuous media. We also survey some problems that complicate the synchronized presentation of this kind of media, and propose some solutions to these issues. In section 4 we present the current operation behavior of CoLab, showing how browsing and continuous media synchronization requests are treated. In section 5 we present the current state of implementation of our platform and we explain its operation. Finally in section 6 we draw some conclusions and discuss future work.

2. Overview of the Collaborative Web Browsing Field

The Collaborative Web Browsing area has taken a
growing relevance in the last years due mainly to its potential for creating new ways of cooperation among Web users. Accordingly several people in both, research and commercial fields have developed different models and implementation proposals aiming to solve this issue. We have made a detailed state of the art in this research area, leading us to the following references which are the most representatives:

- CoBrow [3]
- Let’s Browse [5]
- PlaceWare [10]
- WebCT [11]
- WebSplitter [7]
- E-CoBrowse [4]
- NetDive [9]
- PROOF [6]
- WebEx [12]
- WikiWikiWeb [8]

All these proposals provide alternatives for collaborating on the Web, including some kind of Web browsing synchronization. Some proposals are based on an indexing mechanism which allows the creation of logical neighborhoods of users with similar interests, facilitating then the communication among users. For commercial systems, there are collaborative integrated environments offering users a set of communication tools that allow collaboration.

The main contribution of our approach is the fact that we focus on the dynamic creation and release of synchronization relations among the users. The creation of synchronization relations, as we said in the introduction, leads to binding the Web browsing of a user to that of another user. Synchronization relations are created and released by using some predefined synchronization operators which will be explained later in this paper.

Browsing actions from the users will get synchronized following the previously created synchronization relations. Whenever a Web page contains an embedded presentation, it will follow the same synchronization scheme as that of the page where it is contained.

At the present time there is no collaborative Web browsing application offering such a synchronization scheme, as well as the synchronization of continuous media presentations embedded in Web pages.

3. The Architecture of CoLab

In Figure 1 we present graphically the architecture of our Collaborative Web Browsing system. We can see the main components of our model. These are the Session Manager, the Broker, the Browsing Manager and the MediaSync Manager at the server side, and the Browsing and Media controllers at the client side.

![Collaborative Web Browsing Architecture](image)

**Figure 1. Collaborative Web Browsing Architecture**

The Session Manager is in charge of managing the Collaborative Web Browsing session itself. It is responsible for tracking the connected users’ activities, as well as of the existing synchronization relations. It is also responsible for the eventual integration of the system via the Integration API, with other systems or additional modules.

The Browsing Manager is in charge of all the tasks related to the retrieval of the requested resources by the users. This is not directly accessed by the users, but though the Broker. The reason is that browsing requests are not to be systematically satisfied, but this depends on certain conditions verified at the level of the Session Manager. In this way, when a browsing request arrives, the Broker asks the Session Manager to verify whether or not it should be satisfied.

Finally, the MediaSync Manager takes in charge all the tasks related to the presentation control of the eventual continuous media presentations embedded in Web pages.

3.1. The Browsing Manager and the Broker

The Browsing Manager implements three main functions: a local cache module, a retrieval module, and a translation module.

The local cache module corresponds to the implementation of a basic cache system, which is mainly used, but not only, to satisfy requests coming from synchronous users.

The translation module is responsible for modifying on the fly every retrieved Web page. This is due to the fact that these resources must be “translated” before sending them to the users’ browsers in order to allow our system to track the browsing actions. This translation is also necessary to include into them the necessary controls to be able to synchronize the embedded media presentations. The translation consists mainly in adding some control parameters specific to CoLab to each hyperlink definition.
in the retrieved Web pages. Through the use of these parameters CoLab will be able to detect the browsing actions performed by the users. Moreover, when the Web page has embedded media presentations, this module modifies the HTML code in order to detect plug-in state changes and to notify this to CoLab.

The retrieval module is responsible for retrieving every requested resource. They can be retrieved directly from a Web server or from the local cache module. In the first case the retrieval module sends it to the translation module in order to modify it before sending the response to the user’s browser. In this case, once the resource has been translated, the result is sent back to the retrieval module as well as to the local cache module.

3.2. The Session Manager

The main component of the Session Manager is a synchronization module, which is in charge of treating all the synchronization actions, and guaranteeing the overall consistency of the synchronization state. In fact this component implements the core functionalities of our Collaborative Web Browsing proposal. For the representation of the synchronization state we use a tree structure called SDT (Synchronization Dependency Tree) [1]. In a SDT the nodes denote the connected users, and the arcs denote the synchronization relations currently existing among them. For example for any pair of nodes A and B, if node A is the father of node B, then the browsing actions of user B are synchronized to those of user A. The root node of each SDT represents an asynchronous user. The other users belonging to the same SDT are called synchronous users. The basic notion of SDT is presented in Figure 3, where we see the synchronization relations existing among four users.

![Figure 3. Basic notion of SDT](Image 358x389 to 430x410)

As we have said previously, in its basic operation mode there are two synchronization operators that allow the creation of browsing synchronization relations: I_Follow_You and You_Follow_Me. Given that a single SDT node may have several children, the You_Follow_Me operator can be applied to a single user as well as to a set of users.

Whenever any of these two operators is applied, an authorization protocol is started in such a way that the user whom the invitation was sent to is asked whether or not he accepts it. The synchronization relation is only created if this last user accepts the invitation. Synchronization relations are released by using the operator I_Leave_You, which is unconditional: any of the involved users can apply it, and it will always succeed.

Whenever a synchronization relation is created, the involved users browsing activities get synchronized, and whenever the browsed Web pages contain references to embedded continuous media, the corresponding tools will follow the same synchronization behavior.

3.3. The MediaSync Manager

The MediaSync Manager is in charge of managing the streaming presentations embedded in Web pages. Its main function is to guarantee the synchronization of audio/video presentations for all synchronous users, where the media data can be streamed or downloaded to them.

3.3.1. Media Synchronization Problems

Streaming is a technique developed for transferring data between two computers in a continuous stream. Typically, streaming technologies are fashioned on a client-server model as shown in Figure 4.

![Figure 4. Streaming](Image 432x399 to 503x419)

The client is a plug-in installed in the browser or an external player (for example, RealPlayer, QuickTime, Windows Media) that presents video and audio to the user. It offers normally a control bar with regular media control buttons like play, pause, rewind, fast-forward, position slider, stop and volume control. The client implements a delay-equalizing buffer in order cancel the network jitter and enables the client to sustain momentary drops in the sending rate by playing out of its buffer at a higher rate than the server is currently sending.

The server transmits the data to the client. The server can be a simple Web server (using the HTTP protocol) or a multimedia streaming server, as Darwin Streaming Server [13] or Helix DNA Server [14], where the standardized protocols are RTSP used for controlling the data streaming, and RTP used for transmission of real-time data such as audio and video.

In general, synchronization of audio/video presentations at different destination is not required, but in Collaborative Web Browsing it is an important feature. The characteristics of servers, clients and networks make it difficult to guarantee the synchronization between
presentations at different clients.

This can be most easily understood by considering a hypothetical situation in which two users (for instance, A and B in Figure 4) click on the button “start” at the same time. The factors preventing that the presentations in these two clients are synchronized are: different response times at the click of button “start” in the clients; different connection establishment delays; different transmission delays of the RTSP packet from the client to the server; different response times in the server; different transmission delays of the streaming; different initial buffering times at the clients (often 30 seconds or larger).

Even if the presentations begin at the same time, the temporal non-determinism of the network would avoid this synchronization during the presentations.

The playback control also makes this synchronization difficult. The action accomplished by a user (e.g. Pause in a certain video frame) should be accomplished in the other participants in a relatively short period of time. Moreover, there are interactions that demand new connection requests and buffering whose time of accomplishment cannot be controlled.

3.3.2. Media Synchronization Mechanism in CoLab

The MediaSync Manager guarantees the synchronization of all the presentations by forcing the same presentation state in all synchronous users other than that of the asynchronous one. This module maintains the current state of each continuous media presentation in the session based on State Change messages (old state, current state and presentation position) sent by the Media Controller and it controls the presentations states by sending Playback Control Messages to the synchronous users. Such messages contain the playback action and the presentation position in the asynchronous user at the state change instant.

The playback actions are PrepareToPlay, Play, Stop and Pause. When the Media Controller receives the PrepareToPlay message, it pauses the presentation at the position sent as parameter. When the presentation in each client reaches the paused state, the Media Controller sends a message ReadyToPlay to the MediaSync Manager.

The PrepareToPlay command allows us to control the synchronized start of the whole presentation in spite of the nondeterministic times of connection and buffering. The messages changed when the asynchronous user clicks on the “play” button are:

- after few seconds (around 4s in RealPlayer), the player starts the connection establishment with the streaming server. At this moment, the Media Controller detects the state change and sends it to the MediaSync Manager;
- the MediaSync Manager sends a PrepareToPlay to all participants, including the synchronous user;
- when MediaSync Manager receives the ReadyToPlay message of all participants, it sends a play command starting the synchronized presentations. A temporal control is implemented in order to avoid deadlocks.

This operation to get the synchronized start of presentations happens every time that the presentation in the asynchronous user arrives in the Connecting or Buffering states.

3.4. Browsing and Media Controllers

The Browsing Controller and the Media Controller are two modules present at the client side.

The Browsing Controller is the component in charge of establishing communication with the CoLab proxy server. Through this users’ browsers receive the commands to display Web pages whenever they are synchronized with another user. It contains also all the necessary synchronization controls to allow the user to create and release synchronization relations.

The Media Controller controls the continuous media presentation in the current Web page for all users, in order to get their presentations synchronized:

- it records the state of each audio/video presentation in the current Web page (if any);
- it captures all state changes of the continuous media presentations (caused by a user interaction or an internal event in the plug-in). Depending on the role information and the old and current state of the presentation, the Media Controller only communicates the state change to the MediaSync Manager or executes a treatment related to the received event;
- it receives playback control messages from the MediaSync Manager and executes the playback control.

For synchronous users, the Media Controller avoids any playback control action from the user. In this case, the playback control is done by the MediaSync Manager via Playback Control Messages.

For an asynchronous user, the Media Controller sends all presentation state change to the MediaSync Manager which executes all the necessary actions in order to synchronize the presentations in synchronous users.

4. Operational Behavior of CoLab

Aiming at graphically illustrating the operational behavior of our proposal we present in Figure 6 the case of a typical browsing action performed by a user, and its resulting synchronization with another user.
When this module receives State Change message from all users a (24 and 25) indicating the readyToPlay state, it send to all users the “play” playback control (26 and 27).

5. CoLab’s current implementation

At the present time we have developed CoLab version 2.0, which implements almost all the concepts presented in this paper. It has been implemented on a PC with the Linux RedHat 7.2 OS. The software choice for developing CoLab consists of the Java™ 2 SDK Standard Edition release 1.3.1_13, Jakarta™ Tomcat release 3.3.1a for the Servlets/JSP technology, and JSDT release 2.0 for the CoLab’s internal communication facilities. On the browser side the only technical requirement is that it supports Java™ Applets, and that it can implement the Automatic Proxy Configuration (APC) facility.

In the case of the Media Controller module, it works only with Netscape browser and RealPlayer. The RealPlayer was chosen because it allows accessing the methods that set and retrieve presentation attributes, control the clip playback, and handle user interactions.

The CoLab session window has two frames: one containing the Control Frame, and the other containing the browsing area, where the browsed pages are presented. A screen capture of the browsing and synchronization controls is presented in the Figure 7.

Figure 6 presents also the operation behavior in the playback control of continuous media presentations. For instance, once a Web page containing a continuous media presentation is loaded, if the asynchronous user clicks “play” button, the Media Controller sends the state change (21) to the MediaSync Manager. The MediaSync Manager module updates the presentation state and sends to all users the prepareToPlay playback control (22 and 23).
also see that the user *courtiat* is currently synchronized with user *valentim*. Concerning the synchronization controls we can see that they are divided in two sections: *User Synchronization* and *Role Synchronization*. The first one contains the buttons representing the synchronization operators that can be applied to single users, and the second has only the buttons corresponding to the synchronization operators that can be applied to the roles.

Concerning the control of the embedded continuous media presentations, the playback controls are presented inside the Web page itself. Their behavior is controlled following the currently existing synchronization relations among the users.

### 6. Conclusions and future work

In this paper, we have defined a general-purpose Collaborative Web Browsing system, which provides a new paradigm since it offers the possibility of easily creating and releasing browsing synchronization relations. We think that this orientation gives the users a lot of flexibility for establishing collaboration relations while browsing, creating in this way an environment where collaboration is greatly facilitated.

The current operational implementation, supporting only a subset of the whole functionality, has been developed and it is currently operational. Given that the only information that is exchanged between the connected clients and the server consists only of short messages associated with the synchronization protocol, and URLs to achieve the synchronization of the browsing activities, and that the synchronized resources are directly retrieved from the local cache system, we can affirm that there is practically no overload associated to the operation of the system itself.

We have tested the performance of the synchronization of the browsing activities, and we have found that in real terms it is very good (i.e. for 165 synchronous users the average retrieval time of a Web page is about 1.2 seconds).

Concerning the plug-ins, they are platform dependent and they are external to the browser and so it is difficult, if not impossible, to coordinate the activities of the plug-in. Therefore, it is not possible to implement the collaborative continuous media presentations for all plug-ins and each controllable plug-in will demand a specific code. The current version of CoLab supports the playback control of embedded continuous media presentations only for Netscape browser and RealPlayer.

A characteristic of the current implementation of the synchronization of continuous media presentations feature is that synchronizations always happen in the presentation start or in the presentation resume after a playback control (after a pause, stop, seek, fast-forward or rewind). Therefore, after the synchronized presentation starts, the synchronization can get lost due to, for instance, network congestion factors.

In the future we will keep working in the implementation of all the features of our model, as well as identifying new opportunity areas where we can improve CoLab’s capabilities, as the possibility of adding annotations to the browsed resources in order to facilitate the information exchange among the users. We will also start working on the implementation of the *Integration API*, identifying the possible requirements to be satisfied in order to get CoLab integrated with other collaborative systems and tools. Another subject on which we will start working soon is the implementation of the distributed version of our platform (distributed proxies instead of a single proxy) in order to avoid any performance bottlenecks in presence of heavy workload. Moreover, others plug-ins will be investigated and those that allows access to methods that set and retrieve presentation attributes will be incorporated in the CoLab system.

### 7. References


[8] [http://c2.com/cgi/wiki](http://c2.com/cgi/wiki)


[14] [https://helix-server.helixcommunity.org/](https://helix-server.helixcommunity.org/)