Abstract—People spend a lot of time navigating on the web. When moving from one computer device to another, it would be useful to have access to the navigation data produced in the previous web session. In this article, a synchronization service of navigation data, called Browserver, is presented. It is responsible for keeping user navigation information (tabs, history, forms, cookies, etc.) so that it can be recovered from any device connected to the Internet. Finally, Browserver performance is compared to a similar service, according to hardware and network consumption metrics.

Index Terms—Browser; web session migration; browsing data synchronization.

I. INTRODUCTION

Many people are spending most part of their time on web navigation, whether for work, study or leisure. As this activity is performed by means of web browsers, many personal data are generated and stored locally in the machine used for web browsing. Additionally, there is a multitude of computer devices from which people can browse (from mobile phones to common PC), as well as places from which access comes from (home, work, libraries, etc.).

As a result, the quality of experience of Internet users when navigating on the web relies on their interaction with the browser. Moreover, when they switch from one computer device to another, their productivity will depend on the speed they recover personal browsing data, such as bookmarks, forms, history, open tabs and windows, passwords, custom entries to the spelling checker, etc. In order to provide ubiquitous access to those navigation data, this paper proposes a service that stores it in the web for future retrieval, regardless of computer device or operating system. This service is named Browserver [1] (= browser + server).

While Browserver was being developed, similar initiatives emerged on the web, such as Mozilla Weave [2], Opera Link [3] and Xmarks [4]. The difference between these four services resides in the variety of navigation features they synchronize and the browsers they support.

Thus, this article presents the Browserver service and compares its performance with Weave, which was selected as benchmark because it is the most similar service to Browserver among the three mentioned. The intention is not to polarize the analysis, since both services are in constant development, and future versions may include improvements or bugs.

The reminder of this text is structured as follows. Section II describes the Browserver. Section III describes the environment and the methodology used for comparing Browserver and Weave performance, whose results are analyzed in Section IV. Section V present and discuss related work. Finally, conclusions and future work are outlined in Section VI.

II. Browserver Architecture

A. Web browser personal data

According to [5], [6], the web can be described as consisting of two main types of programs: web servers and web clients. The former manages access to hypertext/multimedia documents. The latter, typically implemented by web browsers, retrieves documents from remote web servers and displays them on screen, either in the browser window, or by invoking an external application. A reference architecture for web browsers was proposed by [6] and is reproduced in Fig.1.

Certainly, a given web browser may differ from the reference architecture. Some subsystems may be implemented as a single module, for simplicity, while others may be spread across multiple modules, to promote flexibility [6].

Anyway, it is the Data Persistence subsystem that stores web browsing data in the disk. This data is distributed along many files, depending on the browser implementation and the navigation features offered by the browser, such as bookmarks, browsing history, forms, cookies, passwords, plug-ins, addons (aka complements, extensions or widgets), open tabs, spell checker, quick search toolbar, thumbnails (aka speed dial), site-specific configurations, and browser preferences.

Therefore, to recover a web session after migrating to a new device, the local files that store browsing data from the source device should be stored in some place so that they can be
retrieved later on the target device. The next section explains how the Browserver performs this procedure.

B. Description of Browserver

Fig. 2. Browserver service architecture.

Browserver [1] is a web service that enables the storage and retrieval of web browsing data. By web browsing data we mean the information stored in the Data Persistence Subsystem described in the previous subsection.

The purpose of Browserver is to enable web browsing migration between different devices (or different users on a single device). So, it differs from the crash recovery service offered by some browsers, whose purpose is to restore web session in the same user of the same device.

Browserver 0.4 stores the following features: bookmarks, cookies, open tabs, forms, history and list of addons currently installed. It is implemented as a client–server architecture. An addon has to be installed in the browser (currently only Firefox is supported, as proof of concept) to collect and retrieve the navigation data. On the server side, a repository stores such data in a login/password basis.

Fig. 2 illustrates the basic scenario of Browserver service. A person browses web pages using any device (source device). Meanwhile, in background, the Browserver Addon synchronizes the navigation data that this person is creating or deleting (steps 1 and 2 of Fig.2). For some reason, that person switches to another device (target device) in order to continue web browsing (step 3). At this time, the user connects securely to Browserver service, after which Browserver synchronizes the navigation data (step 4), promoting a real continuity of the browsing experience.

III. PERFORMANCE TESTS

Performance tests were carried out to compare Browserver and Weave. The latter was chosen as a comparison basis because it also supports Firefox and synchronizes almost the same navigation features as Browserver. Similar services were not included in this evaluation either because there is little intersection of synchronized features (Xmarks), or because Firefox is not supported (Opera Link). Below, the performance metrics, test scenario and experiment procedures are described.

A. Performance Metrics

The performance metrics used in this work can be divided into two major groups: hardware consumption metrics and network consumption metrics. With respect to hardware, memory consumption (in megabytes) and CPU consumption (in %) were measured. With respect to network, we measured the number of HTTP request messages of type GET, POST and PUT; the size of messages sent by the addons (in bytes) during navigation data storage; the size of received messages by the addons (in bytes) during the web session recovery; and the transmission time of messages (in seconds).

B. Tests scenario

The test scenario consisted of only one notebook connected to the Internet. To emulate the user migration from one device to another, we used two different user accounts on Linux operating system Ubuntu 9.04. The source device was emulated by userA account and the target device was emulated by userB account, as illustrated in Fig.3. As each user account saves personal navigation data in different areas of the disk, Browserver and Weave treats the new session as it was in a different machine.

Firefox version 3.5.5 was installed in the notebook. Only three extensions were added to it: HttpFox 0.8.4 [7], for measuring the HTTP traffic, Browserver 0.4 and Weave 1.0b2. The latter two were separately added to the browser, according to the service under test, in order to prevent any mutual interference. Both services have been configured for automatic saving of browsing data.

C. Test procedures

At userA, the following procedures were performed for each service:

- A1 Creating a new user account;
- A2 Login to the service;
- A3 Execution of the task list (Fig. 4);
- A4 Saving of browsing data generated by the task list.

At userB, the following procedures were performed:

- B1 Login to the service;
- B2 Recovery of browsing data.
Browsing data saving (A4) and recovery (B2) were individually performed for each of the following situations:

- only open tabs;
- only bookmarks;
- only history;
- only forms;
- all four features together;

The four navigation features – open tabs, bookmarks, history and forms – were chosen because both Browserver and Weave support their synchronization, providing a common basis for comparison (see Table II). Network consumption was measured for each of the five situations listed above. Hardware consumption was measured only for the latter situation, when the four resources were saved (or restored) together.

The task list described in Fig. 4 consisted in browsing diverse web sites, chosen to compose a variety of services commonly accessed by a typical user: web search, blog, news, video, e-commerce, and FTP download. Each URL address was opened in a different tab within the same browser window. We chose three types of addresses to bookmark: main address of the domain; secondary address within the same domain, reached by clicking; and secondary address within the same domain, typed in the location bar.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>Visit <a href="http://www.google.com">www.google.com</a> (bookmark 1), and search for “synchronization” (bookmark 2).</td>
</tr>
<tr>
<td>Task 2</td>
<td>In a new tab, visit <a href="http://scholar.google.com">http://scholar.google.com</a> (bookmark 3), click on “Advanced Scholar Search” (bookmark 4), fill the fields with the following information, and click on “Search Scholar” button (bookmark 5): with all of the words: “browser state repository service” with the exact phrase: “repository service” with at least one of the words: “browser” without the words: “mining” author: “Song” publication: “Springer” date: “2000” – “2009”</td>
</tr>
<tr>
<td>Task 3</td>
<td>In a new tab, visit <a href="http://www.bbc.co.uk">www.bbc.co.uk</a> (bookmark 6), and click on “Technology News” (bookmark 7).</td>
</tr>
<tr>
<td>Task 4</td>
<td>In a new tab, visit <a href="http://www.bbc.co.uk/art/art/">www.bbc.co.uk/art/art/</a> (bookmark 8), and search for “Byron” (bookmark 9).</td>
</tr>
<tr>
<td>Task 5</td>
<td>In a new tab, visit <a href="http://www.abebooks.com">www.abebooks.com</a> (bookmark 10), search for the book “Data Communications and Networking” and click on the product icon (bookmark 11).</td>
</tr>
<tr>
<td>Task 6</td>
<td>In a new tab, visit <a href="http://www.abebooks.com/docs/Bookstores/">http://www.abebooks.com/docs/Bookstores/</a> (bookmark 12).</td>
</tr>
<tr>
<td>Task 7</td>
<td>In a new tab, visit <a href="http://www.youtube.com">www.youtube.com</a> (bookmark 13), search for “notebook”, and open a new tab by clicking on the first link (bookmark 14). Post a comment.</td>
</tr>
<tr>
<td>Task 8</td>
<td>In a new tab, visit ftp://ftp.freebsd.org/ (bookmark 15).</td>
</tr>
</tbody>
</table>

Fig. 4. Task list.

IV. RESULTS ANALYSIS

This section presents and discusses the results of the comparison tests. For a better understanding, it was split in two parts: hardware consumption and network consumption.

A. Hardware consumption

For each service (Browserver and Weave), hardware consumption was analyzed by means of two metrics: main memory consumption and CPU consumption. At userA, which emulated the source device, those parameters were measured at the following moments:

- Not connected – browser is open, addon is installed, but no user is connected to the service (before A2).
- Connecting – during the login (A2).
- Connected – after login, but no browsing activity (between A2 and A3).
- Before sending data – after performing the task list of Fig. 4 (after A3).
- Sending data – during browsing data storing (A4).
- After sending data – after storing (after A4).

At userB, which emulated the target device, hardware consumption was measured at the following moments:

- Not connected – browser is open, addon in installed, but no user is connected to the service (before B1).
- Connecting – during the login (B1).
- Connected – after login, but without recovering navigation data yet (between B1 and B2).
- Receiving data – during browsing data recovery (B2).
- After receiving data – after recovery (after B2).

The average memory consumption, taken over ten replications, is shown on Figs. 5(a) (source device – userA) and 5(b) (target device – userB). The 95% confidence interval was lower than 0.1 MB.

Memory consumption of the Weave Addon showed to be higher in almost all situations, except when it retrieves browsing data from the server. This is explained by the fact that Browserver automatically opens, on the target device, all the last tabs accessed on the source device, while Weave keeps only a list of recently accessed tabs on the “History” menu. Although opening multiple tabs is convenient for the user, it consumes more memory. In the other situations, Weave Addon consumes more memory because, as it will be seen below, it exchanges more messages with its server than Browserver.

The average CPU consumption, also taken over ten replications, is shown in Figs. 6(a) (source device – userA) and 6(b) (target device – userB). The 95% confidence interval was lower than 0.1%.

During the moments when CPU consumption was 0%, there was no activity of addons Weave and Browserver: either they had already completed the storing/recovery of browsing data, or they had not been requested to do so. On the other hand, during transition times (the *-ing moments), the addons of both services began to exchange messages with their respective servers, stealing CPU cycles. The differences in consumption between Browserver and Weave are explained by the same reasons exposed above.

B. Network Consumption

Table I presents the comparison results of average network consumption. The first column shows the procedure during which the measurements were taken. Each procedure was repeated ten times. For Browserver, the 95% confidence interval was zero for the number of HTTP requests, approximately 1 byte for the message size, and 0.1s for delivery time.
For Weave, there was a greater variability, so that the 95% confidence interval was 3 for the number of HTTP requests, 6 bytes for the message size, and 0.4s for delivery time.

As discussed in Section III-C, it should be stressed that each action listed in the second column of Table I was conducted independently of another. Therefore, “sending all data” and “receiving all data” actions did not correspond to the sum of all sending/receiving actions (tabs, history, bookmarks, and forms), but to a new action where all these resources were saved/retrieved together.

The Weave Addon showed to exchange many messages with its server, unlike Browserver. This is due to the difference in the heuristics used to autosave the browsing data. In Browserver, saving messages are activated by navigation time (5 minutes by default). In Weave, saving is triggered by event (e.g. opening a new tab). Nevertheless, in extreme cases, both approaches can lead to problems. For example, in Browserver, a heavy user can generate, in 5 minutes, a large amount of browsing data, so that it could be difficult to transmit a large message through the network. Conversely, in Weaver, this same user behavior generates an intense traffic of messages with small payload. For future work, it is desirable to develop an adaptive heuristic, which offers a trade-off between number and size of saving messages.

In Browserver, creating a new account implies automatic login, so there was no posts for “connect” action. In Weave, the HttpFox detected the presence of HTTP request messages for “account creation” and “connection” actions, but it was unable to determine the size of sent messages. With regard to forms, we detected that Weave does not save them individually, but it sends them along with other navigation features.

V. RELATED WORKS

Seminal works on synchronizing web browsing data when migrating from one device to another date back to 2002 [8], under the name web session handoff. However, viable and user-friendly alternatives only began to be offered in the last two years. Basically, two conditions stimulated this trend: increasing dependency on web applications, driven by Web 2.0, and diversity of navigation features and tools, driven
by competition among browsers developers. In the following sections we review the evolution of architectures for saving web browsing data.

A. Synchronizing navigation data

Nowadays web browsers are largely used not only as a simple document displayer, but also as an environment for running desktop-like applications [9], [10]. Such applications, known as webapps or weblications, can be accessed through any browser, from anywhere, anytime, in contrast to their operating system and architecture specific counterparts. They also benefit from the possibility of content sharing and collaboration, in addition to dispensing installation or updates.

In this context, it would be desirable that the user could access browsing information ubiquitously across multiple devices. At the same time the Browserver was developed, three other initiatives were being made available on the web:

- Mozilla Weave [2], an addon for synchronizing navigation information of Firefox and Fennec browsers;
- Opera Link [3], synchronization feature embedded in Opera and Opera Mini browsers;
- Xmarks [4], an addon for synchronizing bookmarks and passwords only. It supports four browsers: IE, Firefox, Safari and Chrome.

The three services above work in a distributed way: a local agent collects changes in browsing data of the navigation features they supported, which are sent periodically to a web server. When the user connects lately to the synchronization service, the addon on the target device retrieves the stored data. Weave and Opera Link requires that the same browser is installed in both source and target device, whereas Xmarks supports four different types of browsers.

With respect to user interaction interface, Weave options can only be accessed through Firefox browser menu. Opera Link offers interaction through the browser menu, or via a browser-independent web interface. However, by using this last option, the user may only query the stored navigation data, but cannot update them. In turn, Xmarks enables interaction either through browser menu or web interface, without limitations.

Table II shows the navigation features that are synchronized by the last stable versions of Browserver (0.4), Weave (v1.0b2), Opera Link (v10.01) and Xmarks (3.4.3). Note that some of these features are browser-specific, such as Opera’s note and personal bar.

B. Web Session Handoff

Seminal approaches to synchronization of navigation data were carried out under different names, such as “web session handoff” and “HTTP session mobility”. In [8] it is introduced the Browser State Repository (BSR) Service, which “decouples association between browser state and a device, in favor of association between browser state and its user”.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Action</th>
<th># HTTP requests</th>
<th>Size of sent messages (bytes)</th>
<th>Size of received messages (bytes)</th>
<th>Sending time (s)</th>
<th># HTTP requests</th>
<th>Size of sent messages (bytes)</th>
<th>Size of received messages (bytes)</th>
<th>Sending time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Creating account</td>
<td>1</td>
<td>103</td>
<td>30</td>
<td>1.2</td>
<td>3</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>A2</td>
<td>Login</td>
<td>0</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>5</td>
<td>---</td>
<td>---</td>
<td>312</td>
</tr>
<tr>
<td>A4</td>
<td>Storing tabs</td>
<td>1</td>
<td>2 759</td>
<td>30</td>
<td>4.4</td>
<td>13</td>
<td>6 769</td>
<td>399</td>
<td>81.9</td>
</tr>
<tr>
<td></td>
<td>Storing history</td>
<td>1</td>
<td>4 733</td>
<td>30</td>
<td>2.3</td>
<td>20</td>
<td>13 133</td>
<td>379</td>
<td>88.6</td>
</tr>
<tr>
<td></td>
<td>Storing bookmarks</td>
<td>1</td>
<td>3 289</td>
<td>30</td>
<td>1.4</td>
<td>23</td>
<td>19 668</td>
<td>629</td>
<td>88.2</td>
</tr>
<tr>
<td></td>
<td>Storing forms</td>
<td>1</td>
<td>957</td>
<td>30</td>
<td>1.0</td>
<td>0</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>Storing all data</td>
<td>1</td>
<td>23 821</td>
<td>30</td>
<td>4.0</td>
<td>39</td>
<td>29 760</td>
<td>1 118</td>
<td>236.1</td>
</tr>
<tr>
<td>B2</td>
<td>Recovering tabs</td>
<td>1</td>
<td>66</td>
<td>1 979</td>
<td>2.3</td>
<td>26</td>
<td>178</td>
<td>22 215</td>
<td>84.3</td>
</tr>
<tr>
<td></td>
<td>Recovering history</td>
<td>1</td>
<td>71</td>
<td>4 618</td>
<td>1.0</td>
<td>26</td>
<td>869</td>
<td>12 297</td>
<td>82.5</td>
</tr>
<tr>
<td></td>
<td>Recovering bookmarks</td>
<td>1</td>
<td>71</td>
<td>3 189</td>
<td>5.5</td>
<td>26</td>
<td>869</td>
<td>12 066</td>
<td>75.4</td>
</tr>
<tr>
<td></td>
<td>Recovering forms</td>
<td>1</td>
<td>67</td>
<td>988</td>
<td>2.0</td>
<td>28</td>
<td>968</td>
<td>13 089</td>
<td>74.6</td>
</tr>
<tr>
<td></td>
<td>Recovering all data</td>
<td>1</td>
<td>66</td>
<td>24 508</td>
<td>2.0</td>
<td>37</td>
<td>2 053</td>
<td>24 825</td>
<td>108.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service</th>
<th>Synchronized features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browserver (0.4)</td>
<td>Bookmarks, Tabs, History, Forms, Addon list, Cookies</td>
</tr>
<tr>
<td>Weave (v1.0b2)</td>
<td>Bookmarks, Tabs, History, Forms, Cookies</td>
</tr>
<tr>
<td>Opera Link (v10.01)</td>
<td>Bookmarks, Thumbnails, Typed history, Notes, Personal Bar, Custom searches</td>
</tr>
<tr>
<td>Xmarks (3.4.3)</td>
<td>Bookmarks, Cookies</td>
</tr>
</tbody>
</table>

TABLE I
COMPARISON OF NETWORK CONSUMPTION BETWEEN BROWSERVER AND WEAVE.

TABLE II
NAVIGATION FEATURES SYNCHRONIZED BY BROWSER, WEAVE, OPERA LINK AND XMarks.
Similarly to Browserver, BSR Service contains two modules: BSR plug-in and BSR repository server. BSR plug-in performs two basic operations: (1) it captures the current state of browser (snapshot) and stores it in the BSR repository in a secure manner, and (2) retrieves a saved snapshot from BSR repository to the browser [8]. As BSR dates before the emergence of tabbed browsing and it was not continued, it supports only a few navigation features: last page that appears on the browser, values entered in forms on the last page, cookies, and browser history for backward and forward pages.

Inspired in BSR, [11] proposed a different approach to perform web session migration. Instead of storing personal data in a web server, source and target browsers should perform a handoff procedure by means of a proxy server. Similar approaches are also proposed by [12] and [13].

To guide our discussion, let us review the classification of the architecture schemes that can be adopted to promote web session migration, proposed by [11]:

- **Client-based scheme.** The code responsible for tracking and storing session information (session hand-off component – SHOC) is close to the client. During the session migration, such data can be sent to the target device by means of a P2P network or a web server. Examples: BSR, Browserver, Weave, Opera Link and Xmarks.

- **Server-based scheme.** The SHOC is placed closed to each web server, so that the session recovery on the target device demands a coordination among the SHOCs installed on different web servers which hosts the web pages seen by the user.

- **Proxy-based scheme.** In this hybrid approach the SHOC is placed in an intermediated proxy between the target device and the web server(s). Examples: the architectures proposed by [11], [12] and [13].

While [12] and [11] propose a specific protocol for web session handoff, [13] makes use of Session Initiation Protocol (SIP), defined in RFC 3261. As pointed by the three proxy-based approaches, one of the shortcomings of client-based scheme is the need for modification of browser source-code. However, they have miss generalized by taken BSR as the only representative to illustrate the client-based scheme.

In fact, [11], [12] and [13] use an addon or toolbar to enable communication between browser and proxy server, similarly to modern client-based approaches, such as Browserver and Weave. When the BSR service was proposed in 2002, there were no concepts such as tabbed browsing or extensions. The emergence of these features has enhanced the benefits of client-based schemes.

Finally, the type of service provided by proxy-based architectures is too robust for a web session recovery. In mobile telephony, for example, it is unacceptable that an ongoing call comes to break, even if temporarily. However, in the case of a common web session, short breaks are allowed (unless, of course, in the case of a browser-based VoIP call). Moreover, the proxy-based architecture requires that all web traffic passes through the proxy server before reaching the browser; otherwise the web session handoff is not possible.

As many browsing information do not pass through the proxy (e.g. bookmarks), [12] has also suggested a modified version of their proxy-based solution with client-based components.

**VI. Final Remarks and Future Work**

This work presented the Browserver service, whose goal is to synchronize web browsing data, independent of device and operating system. During comparison tests of hardware and network resources consumption, its performance showed to be superior in most situations, indicating maturity of implementation. We identified the need for development, in the future, of an adaptive heuristics for automatic saving of navigation data that takes into account the size of messages, the time elapsed since the last save and the network traffic.

In addition to the objective metrics discussed here, subjective aspects, such as ease of use and ease-learning, are important to characterize web services. Therefore, usability tests are being conducted in order to improve the interaction between user and Browserver.

Finally, there is a lack of standardization in nomenclature and data structure of the navigation features offered by various browsers. To tackle this problem, future work should be developed around the creation of an ontology for browsing data synchronization, allowing ubiquitous access to this information across multiple browsers.

**ACKNOWLEDGMENTS**

We would like to thank Prof. Tayana Conte, from Computer Science Department at UFAM, Cristina Araújo, Raoni Novellino and Adriana Penaranda for their support in developing the Browserver; and Fundação de Amparo a Pesquisa do Estado do Amazonas (FAPEAM) for funding this research.

**REFERENCES**


