World Wide Web (2014) 17:1295–1319
DOI 10.1007/s11280-013-0238-0

# Emerging research areas in SIP-based converged services for extended Web clients

**Michael Adeyeye · Paolo Bellavista**

Received: 16 May 2012 / Revised: 11 June 2013 /
Accepted: 20 June 2013 / Published online: 5 September 2013


**Abstract** The convergence of Next Generation Networks and Internet-based rich applications are generating relevant industrial opportunities in the market of mobility-enabled services. Even if this trend is widely recognized, there are still a few industrial-level solutions that effectively support session mobility in a transparent way and with the capability of openly integrating with existing and legacy applications. In this paper we propose a SIP-based hybrid architecture for Web session mobility that offers content sharing and session handoff between Web browsers. In addition, its technical originality includes integrating a SIP stack into a Web browser, thus offering the advantage of extending a Web browser to act as a SIP client. Lastly, a rich set of control services that prevent abuse of content sharing and session handoff are introduced into the proposed system. The implemented solution uses SIP in a standard way to migrate Web sessions between Web browsers; it is made up of a SIP integrated Web client and a converged (SIP and HTTP) Application Server that can be easily used to enable session mobility in any kind of Web-based application. In addition, the implemented system has recently evolved to a framework for developing different kinds of converged services over the Internet, analogously to what is possible with Google Wave and the existing telephony APIs. Finally, the paper reports the evaluation of the proposed framework and of the employed technologies, together with directions of future work, in terms of both extension to other application domains and exploration of research areas/models that can benefit form the adoption of SIP and Web-related solutions.

M. Adeyeye (✉)
Department of Information Technology, Cape Peninsula University of Technology,
Cape Town, South Africa
e-mail: adeyeyem@cput.ac.za

P. Bellavista
Department of Electronics, Computer Science, and Systems,
University of Bologna, Bologna, Italy
e-mail: paolo.bellavista@unibo.it

**Keywords** Service convergence · Open API · SIP · Session mobility · Google Wave

## 1 Introduction

Convergence is taking place across numerous research, industrial, and application areas; possible notable examples include network convergence, service-convergence, and device convergence. Network-layer convergence is taking place in networks with focus on technology convergence and integration of different heterogeneous solutions. While application-layer convergence is relevantly occurring in the provisioning of telecommunications and Internet services (e.g., for entertainment applications), device-layer convergence is manifestly taking place in hardware and software manufacturing, which has resulted into new handsets and computer equipments at relatively low cost (thus enabling a mass market of users). There are three main technology trends that are influencing the future of the converged telecommunications and Internet industries. They are: IP-based networks, the growth of Web 2.0, and the rapid evolution of devices (with increasing local resources) [39]; these three trends have sequentially led to network convergence, service convergence, and device convergence in the last years. The interplay of the trends will determine the kind of services that will be available in the future. These services are anyway envisaged to be converged services, i.e., services offered over the above converged provisioning scenario. Owing to this technological evolution, Communications Service Providers (CSPs) may soon no longer be the primary market drivers of communications services but increasingly drift towards a role of mediator and change-enabler [40].

By delving into slightly finer details, the first major changes taking place in the communications industry can be seen in the variety of network technologies. These changes are faster network speeds or broadband access, introduction of mobility and development of IP-based architectures. The second major changes for CSPs can be noticed in the developments of Web 2.0, which have enabled CSPs to expose their services, telco-ICT enablers to bring together numerous applications, and people to produce and consume composite services. The combinations of potential new services are nearly limitless. The third major changes taking place in the communications industry are in advancement in devices, which has allowed customers to fully benefit from converged offerings.

In this scenario, standard solutions for session management in open environments, such as Session Initiation Protocol (SIP) [19, 37, 46], are crucial. In addition, Extensible Messaging and Presence Protocol (XMPP), formerly known as Jabber Protocol, is also widely used to achieve interaction between User Agent Clients (UACs) [38], in particular for instant messaging (in instant messaging clients, such as Pidgin and GTALK) and online multi-player gaming. Although XMPP could be integrated into a Web browser to achieve content sharing and session handoff between Web browsers, as better detailed in the following, the usage of SIP in our proposal offers the advantage of having an adaptive UAC in which a Web browser could act as a SIP client for voice call and possibly be extended to support other SIP-related functionalities (e.g., multimedia-oriented ones).

In addition, the future converged scenario requires innovative concepts for networking and service evolution. Openness, self-organization, self-adaptation,

improved flexibility, and hiding management/operation complexities to users and operators are features expected in the future Internet [29]. Researchers and industry experts have recently started to provide some frameworks and service delivery platforms for converged services provisioning by adopting standard protocols, which are becoming reference points for the above scenario, such as XML, SIP, and XMPP. Only to mention a few notable examples, projects that take advantage of SIP extensibility include the Akogrimo Project [50], which involves embedding Web service data in messages exchanged via the Session Description Protocol (SDP). In addition, in an expired IETF Internet draft [66] by Wu and Schulzrinne, two approaches are identified for transferring URLs between Web browsers for session mobility purposes: the first proposed approach is by sending the URL via a SIP MESSAGE method. Another project that exploits SIP extensibility and similar to our proposal presented in the following was carried out by Munkongpitakhun et al. [33]. Two software packages, related to our work and of some impact in the developers community, are Google Browser Sync [15] and Mozilla Weave [4] Both Mozilla Weave and Google Browser Sync are based on HTTP, which does not provide support for Peer-to-Peer (P2P) interaction, pure asynchronous events (available in the case of SIP SUBSCRIBE/NOTIFY or by integrating with other complementary Web technologies such as AJAX), and multimedia sessions. These solutions are distinctively different from our proposal, though all of them tend to improve the Web browsing experience. This paper presents our original proposal in terms of a SIP-based hybrid architecture that leverages SIP, HTTP, and XML to provide converged services. The project is a novel approach to session mobility in HTTP by using SIP as the carrier of session data. It introduces a new service in Web browsing context, namely content sharing and session handoff among Web browsers. The service is derived from SIP Mobility mechanisms—Third Party Call Control and Session Handoff mode flow for transfer to a single device [45]. To avoid possible abuse in the exploitation of our proposed support, the service is controlled by a SIP server, called Converged Application Server (CAS). CAS has the capabilities of blocking, screening, and forwarding content sharing or session handoff requests; moreover, it offers parking and picking up of requests through its session tracking (with history) mechanism. In addition, we propose and implement a new Web browser architecture that integrates SIP to achieve HTTP session mobility and to provide SIP functionalities. In particular, the proposed extension, called TransferHTTP, makes the Web browser act as a SIP client, by enabling i) voice calls from it and ii) Web session transfers to other Web browsers.

The remainder of the paper is organized as follows. Section 2 discusses some common problems experienced on the Internet. Section 3 presents an overview of our original SIP-based proposal for interoperable session mobility support for converged services. Section 4 compares our work with various Web and telephony application development tools and technologies, by highlighting the different targeted objectives and the pros/cons of each solution. After that, the last part of the paper points out the wide potential area of application of the proposed solution, its easy application to these areas, and the future academic-/industry-oriented research directions that we will follow to extend our work in the next months. The comparison with other technologies in this article does not contain experimental evaluation in terms of performance and accuracy, another article [2] written by the authors reported the performance and accuracy of our work.

## 2 Use case scenarios of some common problems experienced on the internet

Below are two of the common problems regularly encountered on the Internet. These problems are described below using Use Case Scenarios.

1. Bob is in a laboratory at school browsing a newspaper Website when he comes across an interesting article that he wants to share with Alice. Alice, his colleague, is in a coffee shop also browsing a different Website, via a wireless access point provided in the coffee shop. Assume that both of them are online on Facebook and Yahoo Messenger, Bob quickly copies the URL of the article and sends to Alice via one of the Instant Messaging services. At the same time, he invites Alice to a voice chat to discuss the article. In a situation where Alice is not online, Bob sends the URL in an email to Alice and asks if they could discuss the article later.
2. Alice, researching on her project and using one of the PCs in a public library, is asked to fill in a form before she can download software she needs for the project. She has only logged in minutes ago and now asked to fill a form that requires a number that she cannot recall offhand. She realizes that she would have to restart the whole processes on her PC when she reaches home. Finally, she quits the Website and tries to do something else until she gets home when she will be able to continue her project.

Scenario one depicts content sharing, and scenario two depicts session handoff. Both share the same use case diagram shown in Figure 1. In scenario one, the referrer is Bob and the referred party is Alice; while in scenario 2, Alice acts as the referrer and the referred party. The problems depicted in the above scenarios are very similar to what individuals face today when surfing the Internet.

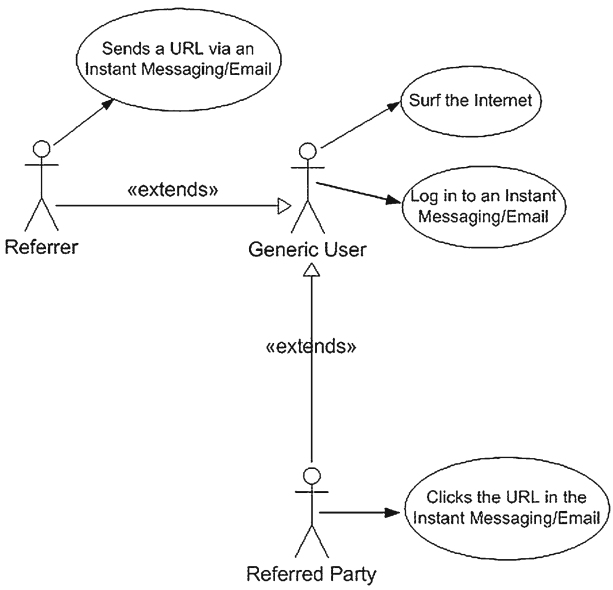


**Figure 1** Use case diagram of content sharing and session handoff

Scenario one shows slow and inefficient ways to referring someone else to view the same Web page being viewed at the same time. This problem is identified as one of the ways the Web browsing experience could be improved if an efficient solution exists. The solution should cater for sending the URL to the intended recipient by a click or not more than two processes rather than copying the URL and using another software or service to accomplish the task. Better still, the solution should also cater for the voice interaction.

In scenario two, referred to as session handoff, the user, Alice, would like to continue filling the form at home without having to log-in again and navigate the Website to the form page. Most times, individuals want to continue viewing the same Web page later and at a different place. A large amount of HTTP signalling is involved moving from one link to the other, and a cost is incurred in the signalling, most notably where Internet access is expensive, though the cost could be small.

## 3 A SIP-based hybrid architecture for converged services in Web session mobility scenarios

Several works have investigated the issue of Web session mobility using client-based, proxy-based and server-based architectural schemes [8, 20, 47], thus demonstrating the research/industrial interest in the topic. Session Mobility enables seamless transfer of a Web session between different devices, based on user preferences and other context data (e.g., availability of a given access network). The reasons for session transfer include cheapest access cost, better user experience, and physical user mobility. As convergence is now moving into the mobile space, there is a strong push from Triple play of voice, video, and data to Quad play of voice, video, data, and mobility services [31].

In our solution, we decided to adopt SIP as the basic starting point technology because its User Agent (UA) could act as both a User Agent Client (UAC) and a User Agent Server (UAS), thereby making two SIP UAs to interact with each other without a mediator or proxy. SIP is also used because it could offer multimedia services between two or more extended Web browsers in both peer-to-peer and client-server architectures. We understand that H.323 could be used over SIP. However, a major difference between SIP and H.323 protocols is that SIP media is independent of its signaling protocol. While SIP is simply used to set up and tear down media sessions, Session Description Protocol (SDP) is used to define, negotiate, and handle the media streams. H.323, on the other hand, specifies in detail which underlying protocols will be used to provide a media service. As a result, SIP is usable in so many areas than H.323. Another difference is that SIP is based on text-based protocols like HTTP and SMTP, while H.323 is written in binary code. This difference makes H.323 makes less friendly to a programmer/developer without significant experience and development tools. Although H.323 is much more mature as a protocol than SIP, it has less interoperable features across multi-vendor equipment. SIP-related service providers can offer outsourced services that were not possible with TDM or H.323 technologies.

HTTP UA could only be a client or a server, thereby making it difficult to have two UAs interact with each other without a server or extending the protocol. In particular, our solution adopts the distributed hybrid-based architecture shown in

Figure 2 for content sharing and session handoff between Web browsers. In the following we will use the term content sharing to indicate the ability to simultaneously view the same Web page on two browsers at the same time, by typically transferring only the Web page Universal Resource Locator (URL). In addition, we define session transfer as the ability to move the whole Web session, including not only the Web page URL, but also all needed session data (cookies, hidden form elements, and rewritten URLs). An example of content sharing is when Alice refers Bob to visit the same news Website that she is browsing: in this case, she would only want the URL to be sent. An example of session handoff, instead, is moving an email session between two devices with the final goal of continuing to check emails, with the same view of read/unread/cancelled messages, etc., without having to sign in again.

The Web browser extension in this project, which is called TransferHTTP, was developed for Mozilla Firefox version 2.0–3.0 and required modifying the architecture of the Web browser. In addition, session data are transferred in plain text format between Web browsers, though recommendations are made on the security of session data. The SIP stack in the TransferHTTP extension was wrapped as a shared library and a new Cross Platform Component Object Model (XPCOM) was written to interact with it. XPCOM makes it possible to write language-agnostic components, thus separating the implementation from its interface [7]. This approach provided a new layer of abstraction to the Web browser in order to integrate the SIP protocol. JavaScript was used in the implementation to pass user data to the XPCOM extension, which was implemented in C++. The TransferHTTP extension added i)

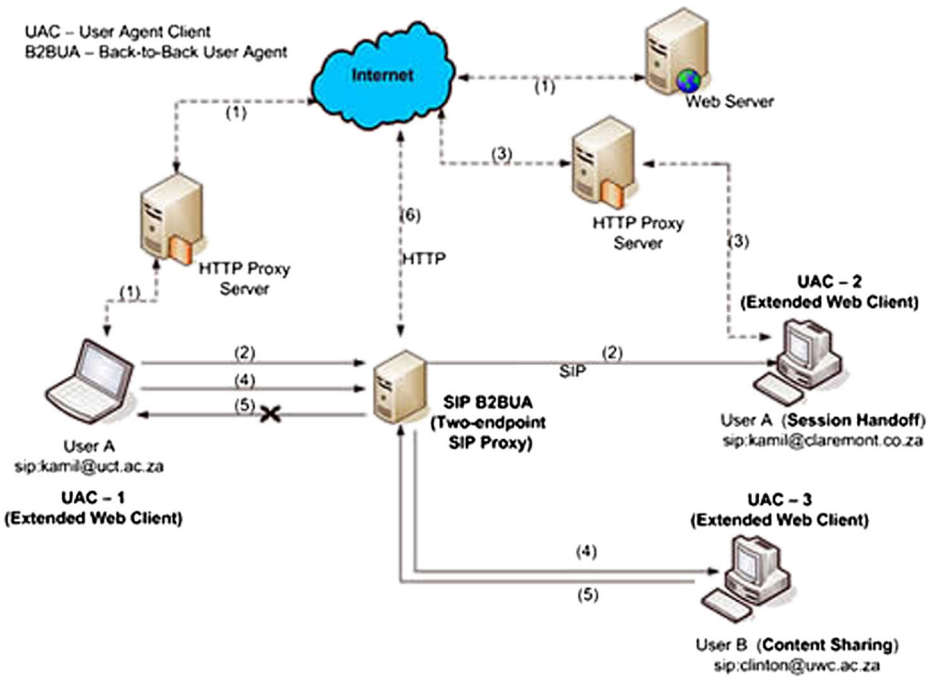


**Figure 2** The hybrid-based architectural scheme

a new XPCOM with the contract id "@ngportal.com/SIPStack/SIPStackInit;1" into the existing XPCOMs in the browser, and ii) an interface, named "IJImyStack", in Interface Definition Language (IDL).

CAS [58] is a SIP B2BUA, which responds to both HTTP and SIP requests. A SIP B2BUA is a call-controlling component that maintains a complete call state and participates in all call requests. It is involved in call establishment, management, and termination. CAS is involved in Web session blocking, screening, and forwarding, by acting also as a SIP registrar. Participating Web clients appear online when they register with CAS. This service could work in either a P2P environment or a client-server model. CAS also acts as a SIP proxy, which is required when the UACs are behind Network Address Translation (NAT) boxes. In addition, CAS is a logical entity with both UAC and UAS capabilities that has full control over traversing dialogs and SIP messages. The Mobicents Communications Platform [14, 53] was used to implement it.[1] Mobicents is an open Java-based platform that enables creation, deployment, and management of services and applications that integrate voice, video, and data across a range of communication networks and devices. It implements and delivers both competing and interoperable programming models—JAIN Service Logic Execution Environment (JSLEE) and SIP Servlets—to develop Web and VoIP applications that work together.

In addition, CAS implements Web session mobility parking and pickup. Although applications could be developed using either JSLEE or SIP servlets on top of the Mobicents platform, our implementation is based on the Mobicents SIP servlets programming model. When the proxy receives a SIP request, the Application Router (AR) in the server is called by a SIP container. The AR selects the appropriate SIP servlet application to serve the SIP request. The SIP servlet application in this work responds to these requests—SIP INVITE, REGISTER, and MESSAGE.

The architecture is a composite system that is made of heterogeous elements - an extended Web client and an application-based SIP proxy (hybrid-based architecture). The interaction in Figure 2 is between extended Web browsers, enhanced with SIP capabilities, and acting as SIP UACs and a two-endpoint SIP proxy that coordinates browser-to-browser interactions. SIP proxy enforces (via SIP, solid arrows in Figure 2) Web session blocking or forwarding by using standard SIP user identities and access policies declared by the users. In the first phases of our project, we designed and implemented a novel service, referred to as content sharing and session handoff between Web browsers, as extensively described in [1]. This implementation has provided a fast and efficient way of referring someone else to the same Web page currently viewed by the referrer rather than the slow way of copying, pasting and sending the URL in a chat session or an email. While [8, 20] broke the IETF RFC 2616 and 2965 specifications in their implementations, this work has provided a way of transferring a stateful web session from one PC to another that does not break the specifications.

Secondly, we have worked on integrating a SIP stack into a Web browser, by offering the advantage of extending a Web browser to act as a SIP client [1]. In this way, Web browsers can act as SIP clients, thereby setting up multimedia sessions between two or more users. Most notably, our extended Web browsers have unique

[1] It is publicly available for the SIP research community at http://transferhttp.berlios.de.

SIP addresses to interact with one another like PCs. Lastly, a plethora of control services that prevent abuse of the content sharing and session handoff service were introduced [3]. The Web browser extension, developed in this work, co-ordinates with the novel SIP-based Converged Application Server, which was also developed in this work, to enable session mobility and prevent abuse of its services.

Since content sharing and session handoff are critical operations, potentially prone to security problems such as malicious users acting as men-in-the-middle between two interacting parties or possible abuses of services offered by the browsers, some features, such as Web session blocking and forwarding, are introduced at the proxy of the system to control the interaction between the browsers. These features are found in telecommunications, where phone calls can be blocked, forwarded or screened. A feature or service can be defined as a value-added functionality provided by a network to its users [16]. Although the features—call blocking, call forwarding and the likes—are specific to telecommunications, they are feasible in the Web-browsing context owing to the interactions between two or more browsers.

The graphical user interface of our extension is shown in Figure 3; it is also a screenshot of the extension, when in use. TransferHTTP adds a new menu "HTTP Mobility" to the menu bar in the browser and a "Preferences" submenu to the new menu. The Preferences submenu, when clicked, gives users the opportunity to configure the browser. The settings include the SIP proxy address/port number and the SIP username. Figure 4 shows CAS User Interface. CAS logs all session

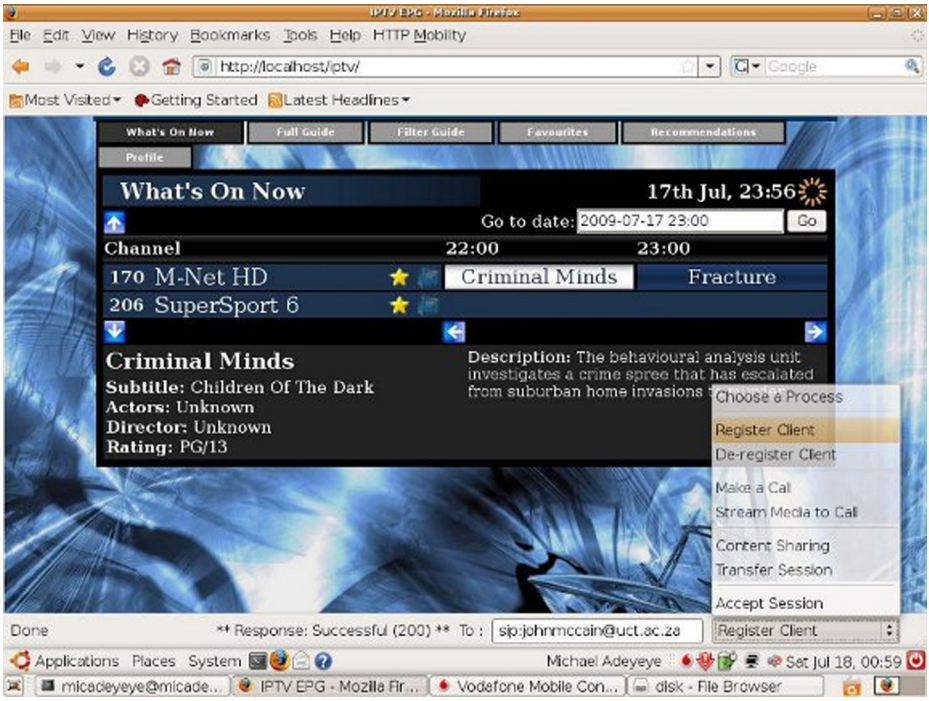


**Figure 3** The TransferHTTP user interface

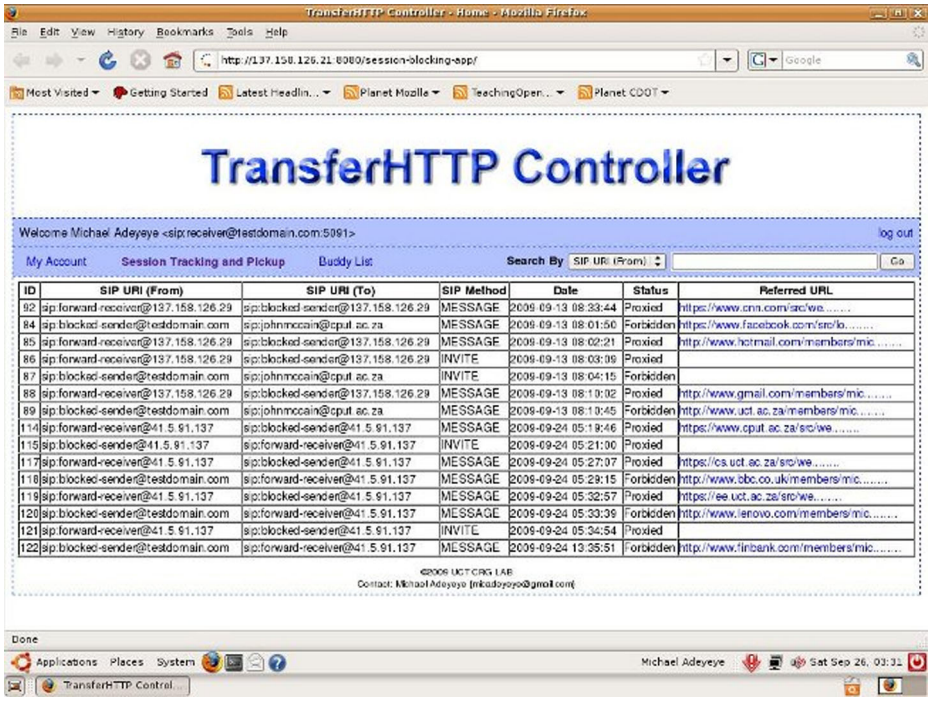


**Figure 4** The TransferHTTP proxy user interface (Session tracking and pickup page)

transfer requests, call setup requests and actions taken on them. It provides the source SIP address, the destination SIP address, the SIP method, date, action taken (also known as status) and the referred URL in the case of a session transfer request. This information is available under the "Session Tracking and Pickup" page, as shown in Figure 4. In addition, the "My Account" page enables a user to set their policies. Information available here includes their SIP URIs with their log-in details and policies. A user could set how every request should be handled. That is, they could set what requests should be blocked and what address should a request go to. They could add/change/delete their identities. On the other hand, the "Buddy List" page contains a list of their contacts. A user could add a new contact via the page and could also check the Buddy List page to see if their contacts are online or offline.

This project leverages three Free Open Source Software (FOSS) packages, namely Mozilla Firefox [54], PJSIP [57] and Mobicents SIP Server [14, 53]. Mozilla Firefox is a Web browser developed by the Mozilla Corporation, and PJSIP is a small footprint, high performance, flexible and Open Source protocol stack. The Mobicents SIP Server is the most popular Open Source Application Server for the Java Platform. The Mozilla Framework is widely used in the academic environment to develop Web and desktop applications. A small footprint SIP stack was used in this implementation. Its size was 2.2 MB, while a typical Web browser installer could be 9 MB in size. The packet size of the SIP stack was extended to 16 kB in order to send session data at once. Session data are transferred in SIP MESSAGE method because an SDP text payload must not be more that 1kB in length [18]. In addition,

**Figure 5** An example of an XML document that can be encapsulated in a SIP message body

```
<xml version="1.0">
<httpsession>
<url>http://mail.live.com/TodayLigth.aspx?FwaD15723</url>
<inputs>
<input name="en" type="text">english</input>
<input name="mOptionsList" type="select-one">3</input>
<input name="_VIEWSTATE"
type="hidden">79gh8</input>
</inputs>
<cookies>
<cookie name="MSPRequ" domain="true" host=".live.com"
path="/" isSecure="false"
expires="121831">3gJeDI</cookie>
<cookie name="PHPSESSIONID">HDYK7DJ6K3</cookie>
</cookies>
</httpsession>
```

SIP INFO method could not be used because it is only used for communicating mid-session signalling information along the signalling path for a call. However, SIP INFO method could be used when a session already exists between two UACs, such as a voice call. Although a SIP PUBLISH could be used in place of a SIP MESSAGE, it is better used alongside a SIP SUBSCRIBE to achieve conference model of Web browsing as discussed in [66].

After successful integration of a SIP stack, the Web browser was subjected to a performance test to determine outcomes when the SIP stack was running as a background service. Here, the outcomes refer to probable changes in download and upload speeds of the browser, as well as its memory consumption. The SIP stack in the TransferHTTP extension was wrapped as a shared library and a new Cross Platform Component Object Model (XPCOM) was written to interact with it. XPCOM makes it possible to write language-agnostic components thereby separating an implementation from its interface [7]. This approach provided a new layer of abstraction to the Web browser in order to integrate the protocol (SIP). JavaScript was used in the implementation to pass user data to the extension's XPCOM, which was written in the C++ programming language.

XML Path Language (XPATH) is a non-XML language for selecting nodes in an XML or an HTML document, and Dynamic Hypertext Markup Language (DHTML) is used to add behaviour to Web experience that HTML 4.0 could not offer. While DHTML involves JavaScript and Cascading Stylesheet languages, Asynchronous JavaScript and XML (AJAX) and Web 2.0 are technologies similar to DHTML in purpose. In this work, a JavaScript code that used XPATH was written to extract all form data in a Web page. These form data are sent alongside the session data transferred between two UACs. On receipt of the data in an XML format as shown in Figure 5, XPATH was also used to extract the necessary data. With regards to the scope of this work, it is an experimental work that does not ensure the security of data. Although a TLS-supported SIP proxy is recommended in the production environment, the proxy used in this project does not have TLS enabled. In addition, the proxy only executes the SIP application developed in this work and responds based on users' policies.

## 4 Evaluation and comparison with state-of-the-art solutions

The browser extension developed in this project is referred to as TransferHTTP, while the SIP-based converged application is referred to as CAS (Converged Application Server). Both the browser extension,[2] which was developed for the Mozilla Firefox browser, and CAS[3] are publicly available for the Web and SIP research communities. This section discusses the comparisons of the work with the new communication and collaborative tools—the Google Wave, WebRTC and existing Open APIs for telephony and converged application development.

### 4.1 Discussing emerging Web 2.0 models and tools

#### *4.1.1 The Google Wave*

Table 1 shows the comparison of this work with Google Wave. Google Wave, under the new name Apache Wave, is a tool for communication and collaboration on the Web. It uses an open protocol [52, 61], so anyone can build their own wave system.

The Google Wave API allows developers to use and enhance Google Wave through two primary types of development, namely Extensions and Embed. The Extensions represent the server-side, while the Embed represents the client-side. The extensions (also called the Robots API) can be developed using the Java Client Library, Python Client Library, or Gadgets API, while the embed, which is embedded into a Web application, is always written in JavaScript.

The Google Wave and this work (TransferHTTP+CAS) provide the same services, though over different architectures. While the Google Wave API is used to develop applications that reside on a Web server, TransferHTTP APIs are used to develop applications that reside at the client end. For example, the Click-to-dial in Google Wave [51] requires the server to set up a call session, while in TransferHTTP, the client sets up the call session. In Google Wave, the robot in the Web server is responsible for the signalling, while in TransferHTTP, the SIP stack in the browser does the signalling.

The Google team has separated the signalling (HTTP and XMPP) in a bid to maintain the current Web architecture. The Extensible Messaging and Presence

**Table 1** Comparison of Google wave and TransferHTTP+CAS

| | Google wave | TransferHTTP+CAS |
|---|---|---|
| Client Technologies | JavaScript in HTML | XUL and JavaScript |
| Server Technologies | Python/Java/ Gadget | Java (HTTP/SIP Servlet) |
| Architecture | Server-based | Hybrid-based |
| Protocol | Wave (Extension to XMPP) | SIP |

[2]Additional information and the prototype code of the TransferHTTP extension are available at http://transferhttp.mozdev.org.

[3]Additional information and the prototype code of CAS are available at http://transferhttp.berlios.de.

Protocol (XMPP) stack [38] in Google Wave resides at the server, and its APIs are written for third-parties to help them develop converged applications. Hence, it could be referred to as a server-based architectural framework for service creation. In this work (TransferHTTP+CAS), a SIP stack is integrated into a browser to provide similar services. It was found out that the integration of a SIP stack into a browser does not impede its performance thereby making this work a viable approach to create converged services. A hybrid-based architectural framework is created in which services could be provided by the client using the TransferHTTP APIs. In addition, a number of services could also be provided by the proxy component (CAS). These proxy services could prevent abuse of the services offered by the client. They are meant to control the interaction between the browsers. While the proxy services could be developed using the Mobicents SIP Servlets and JAIN SLEE APIs, the client services could be developed using the TransferHTTP APIs.

In summary, irrespective of the technologies or programming languages used in the Google Wave and this work (TransferHTTP+CAS), the difference between them is that the Google Wave only has a stack (an XMPP stack) in its server, thereby making it a server-based architectural framework for service creation, while TransferHTTP has a stack (a SIP stack) both in its client and proxy, thereby making it a hybrid-based architectural framework for service creation.

#### *4.1.2 WebRTC*

WebRTC is an open framework that offers Web application developers the ability to write rich real-time multimedia applications (e.g. video and gaming applications) on the Web without requiring plugins or extensions. Its purpose is to help build a strong Real Time Communication (RTC) platform that works across multiple Web browsers and platforms. In an implementation, the WebRTC API will abstract several key components for real-time audio, video, networking and signal [22, 63].

One of the IETF RTCWEB WG [21] is currently discussing how to integrate WebRTC with deployed SIP equipment and domains. An area of its application is communicating from WebRTC applications to existing deployed SIP/RTP-based Voice/Video-over-IP devices at the signalling and media planes. It may require an interworking middlebox function (e.g. an integrated Web Server module) in the media-plane. However, the deployed devices should communicate using SIP at a signaling layer rather than HTTP. Other protocol implementations, such as XMPP and H.323, can also be achieved.

From the industry perspective, the Web browser software industry is also implementing browser-to-browser interaction in various ways. Although WebRTC is currently being standardized, it is however possible that some of its implementations require extending an existing terminal (like a Web client in our work), a proxy or a server.

#### *4.1.3 Open APIs*

Open standard APIs are desirable for introducing new services because they make the separation between an application and its platform explicit. They allow application portability and allow the functions of the platform to be used by multiple applications easily. APIs allow programmers to focus on the logical flow of applications using only the necessary functions provided by the platform, rather than concerning

themselves with low-level details of messages that must flow across the network. As a result, a well-defined API allows the application programmer to work at a higher level of abstraction than that of the protocol [23].

The SIP API reflects the SIP protocol fairly closely [5, 48, 49]. It is useful for situations where the application is rather simple and where the underlying network is known to be an IP network. However, the SIP API is at a lower level of abstraction than the call control APIs, such as JTAPI, JAIN, and Parlay APIs. As a result, it offers the programmer finer grained control, better performance than the call control APIs.

Another messaging and presence protocol that is widely used is Extensible Messaging and Presence Protocol (XMPP). Its APIs have been used to develop XMPP clients, such as Google Talk and Pidgin. It is gaining wide acceptable in the software industry, where it is being used to develop communication and collaborative tools. Examples of shared applications built with XMPP are shared whiteboard and chessboard [64]. Another work that is currently exploiting XMPP is Google Wave, which has just been discussed.

The need for Open APIs is greatly increasing [25, 32, 42]. The APIs are needed for user-generated services. Although there are APIs, such as Google APIs and Parlay-X APIs, for developing Web 2.0 applications and basic Web service APIs for access to circuit-switched (CS), packet-switched (PS), and IMS networks, they fall short of enabling innovative converged applications or services from users.

The Google Wave project currently looks promising for application developers but could be limited in functionality in the near future. The Parlay-X APIs are already claimed to have very limited functionality [32]. The reason is that they are not designed to handle the data model for the entire service or signalling in Telecommunications.

This work (TransferHTTP+CAS) presents APIs that expose the signalling in Telecommunications to create innovative applications. The APIs make it possible to create applications that can use the instant messaging and presence features in SIP [36]. The TransferHTTP APIs currently expose the SIP REGISTER method so that a browser can register with a SIP network. They also expose the SIP MESSAGE method to send messages or chat and the SIP INVITE method to make calls between two browsers. In addition, there is a media-to-call function that could be used to create a media broadcast service.

On sample applications that use these APIs, the TransferHTTP APIs have been used to implement HTTP session mobility service. In addition, its media-to-call interface has been used to create a media service. The APIs are available for use when the TransferHTTP extension [59] is installed in the Mozilla Firefox browser.

The APIs, which were released under the Mozilla Public License (MPL), can be extended to support other SIP methods, notifications or functionalities, and the technologies used in creating the extension are already shown in Table 1. The underlying component, which was written in C++, is scriptable; that is, a user could develop a JavaScript application that implements the methods and arguments.

To enjoy the full potential of these APIs, developers are advised to develop applications based on XML User Interface Language (XUL); XUL was the language used to develop the Mozilla Firefox user interface. Hence, a XUL-based application developed by any interested user will be able to use the APIs in the web browser and the extension, unlike Web applications, which are restricted for security reasons.

### 4.2 Evaluation of this work (TransferHTTP with CAS) with other Web session migration works

Table 2 shows the comparison of this work (TransferHTTP+CAS) with other existing Web session migration approaches. Here, TransferHTTP refers to the extended Web browser. Although Canfora et al. [8] and Hsieh et al. [20] are both proxy-based architectural schemes, only Hsieh et al. is used in this evaluation. Hsieh et al. is chosen because it has more functionalities, such as support for optional client program, than Canfora et al.

Works compared with this work (TransferHTTP+CAS) are Browser State Preservation and Migration (BSPM) [47] and Stateful Session Handoff for mobile WWW [20]. BSPM is a client-based architectural scheme while mobile WWW (that is, Stateful Session Handoff for mobile WWW) is a proxy-based architectural scheme. Our work is based on a hybrid architectural scheme that uses SIP for Web session migration.

Both BSMP and mobile WWW provide session handoff between two Web browsers. This work (TransferHTTP with CAS) also provides session handoff. Session handoff in the three works requires that all cookies are sent to the Web browser, though the cookies are sent through different mechanisms.

This work, like other works, offers content sharing, which entails one Web browser referring another Web browser to have access to the same Web resource. All of these works however have more differences than similarities. Regarding modifications made to Web browsers architecture, BSPM is a client-based architectural scheme that requires modifying the architecture of a Web browser. TransferHTTP with CAS, which is a hybrid-based architectural scheme, also requires modifying the architecture of a Web browser. On the contrary, mobile WWW is a proxy-based architectural scheme that does not require modifying the architecture of a Web browser. It however supports an optional client program that modifies the architecture of the Web browser when the client program is installed.

BSPM lacks user-client interaction while mobile WWW provides user-client interaction when its optional client program is installed. TransferHTTP with CAS, like mobile WWW, also provides user-client interaction. User-client interaction offers a user the ability to continue a task, such as filling a form, rather than starting afresh at another end after a session handoff.

In terms of session registration/tracking, BSMP offers only user authentication, which is used to register a session. Mobile WWW however offers session tracking in addition to user authentication. TransferHTTP with CAS also supports user or client authentication. History handoff is a feature found in all of the work. The history is accessible in this work via CAS; it is available under session tracking, which is a secondary service offered by CAS.

Lastly, TransferHTTP ver. 1.0 could work in a P2P environment without a SIP proxy server (CAS). In addition, it introduces a new Protocol, SIP, into a Web browser that makes it possible to set up voice call with the destination Web browser. This offers a new way of collaboration in the Web-browsing context. Unlike other approaches, it does not break the HTTP security rules. It was ensured during the implementation that no two interacting Web browsers would have the same session data, most notably cookies and hidden input elements, during a session transfer.

**Table 2** Comparison of TRANSFERHTTP+CAS with other existing Web session migration approaches

| Item | Approaches | | |
|---|---|---|---|
| | BSPM | Mobile WWW | TransferHTTP+CAS |
| Modification to user devices | Yes, substantially modified browser | No, but an optional small client program that modifies browser can be installed. | Yes, a small client program (called TransferHTTP) that modifies browser is required |
| Session registration/ tracking | Supports session registration (user authentication); No tracking | Supports session registration via the client program and User Agent Proxy (UAP). Session tracking is achieved by accessing the UAP's Web pages | Supports session registration (user authentication) via the client program and UAP(CAS). Session tracking is achieved by accessing CAS Web pages |
| Tracking session data (cookies, form data, encoded URL) | Unnecessary (not implemented) | With the aid of UAP, but unable to track unsubmitted form fields when client program is not installed. | Tracks session information via the small client program |
| Handoff action | Source saves to BSPM repository and then Target retrieves from it. | UAP retrieves unsubmitted form fields from source via small client program. And target retrieves from the UAP. | Source sends session information (unsubmitted form fields inclusive) to Target via UAP (CAS). |
| History and cookie handoff | History is loaded into browser And all cookies are sent to it. | History is sent as a list of Web pages. UAP sends necessary cookies to browser. | History is accessible via CAS. All cookies in a request are sent to browser. |
| Web pages handoff | Content is directly loaded into browser. | Without small client program, UAP redirects browser to open URLs. With small client program, UAP sends URLs to client program and browser opens the URLs. | UAP (CAS) sends URLs to client program, and browser interacts with client program to open URL. |
| Summary | A client-based scheme i.e. a client is modified. Needs a repository that stores session data. No user-client interaction. It only introduced introduced session handoff. | A proxy-based scheme i.e. A proxy is required to perform the session transfer process. An optional client program can be used to support user-client interaction. And it only session handoff. | A hybrid-based scheme. Introduced SIP into the client and can use an optional SIP proxy (CAS).CAS is not needed in Peer-to-Peer. It supports user-client interaction. In addition to session handoff, it introduced voice call, content sharing, session mobility blocking, forwarding, screening, parking and pickup. |

## 5 Areas of application of TransferHTTP

1. **Improving access to Multi-channel and Multimodal Applications**
   A multi-channel application presents its content to the end user based on their connecting device or user agent. In a multi-channel access, enterprise data and applications are accessible from multiple channels. Multimodal access, on the other hand, is the ability to combine multiple channels in the same interaction or session [13]. Numerous works have already explored Web and dialogue system convergence for multimodal and multi-channel services over converged networks [6, 12, 28].
   Both multi-channel and multimodal applications require a VoiceXML gateway to interpret VXML pages and access the voice part of the applications. The building blocks of a VoiceXML gateway include Text-to-Speech (TTS), Automatic Speech Recognition (ASR) and VXML Browser [26]. A multimodal browser, such as Opera multimodal browser [56], is always required to access a multimodal application. With more VoiceXML gateways integrating SIP, a SIP-integrated Web browser could use its embedded SIP functionality to provide the required voice interaction with a VoiceXML gateway. In this case, by adopting our proposal, there would be no need to install a multimodal browser. Hence, the SIP integrated Web browser could be used to access a multi-channel/multimodal application or SIP applications in the endpoints.
   Both directed-dialog, which requires Dual-Tone Multiple Frequency (DTMF), and Interactive Voice Response (IVR) interactions could be achieved with a SIP-integrated Web browser; examples of VoiceXML gateways that already have supported SIP are VOXEO [60] and OmniVox3D [55].
2. **Modifying How Click-to-dial Works**
   Integrating a SIP stack into a Web browser could modify or extend how click-to-dial applications work [23]. When a "mailto:" command in a hyperlink is clicked, it launches a mail client in order to compose a mail to the email address in the hyperlink. Having standardized the command "tel:" in a hyperlink [43], when such a hyperlink is clicked, it could initialize the built-in SIP stack in a Web browser and set up a call between a phone and the Web browser. In click-to-dial, an application server is always responsible for establishing a call session between two phones.
   With a SIP stack integrated into a Web browser, a call session could be established at the client end. In addition, integrating a SIP stack into a Web browser could make it possible to transfer a call from a mobile phone to a PC, when the caller or callee moves to an environment that has poor signal strength but a very fast Internet connection. Although a separately installed SIP client could offer this service, a SIP-integrated Web browser would be better, most notably when the person wants to browse the Internet at the same time. In addition, a SIP-integrated Web browser encourages adaptive UAC, whereby the web browser could be used as a SIP client.

## 6 Current/future directions of extension and application server

The research results and the framework presenetd above are of wide interest and applicability in many application domains and in many deployment cases, as already

shown in a few examples in the previous parts of the paper. Given this wide applicability and general-purpose relevance, we are currently working on extending our framework along several directions, which we have structured in the following, also for the sake of better readability, in the two macro-categories of industrial research directions (short-term extension work, with expected results of high impact on both the market and the practical application of our framework) and of academic research directions (medium-term extension work, including basic and modeling research, with expected results of strong originality).

### 6.1 Industrial research directions

Below there is an extensive list of industrial research directions that are promising for improving the applicability of our project.

1. **Multi-domain Implementation**
   Although the proposed architecture was implemented using a SIP server (i.e. over a single domain), it could be extended to support multiple domains or servers. That is, the implementation or deployment of this reference system could involve multiple CASs. To achieve this, the SIP application could be deployed on a Service Delivery Platform (SDP) that supports multi-domain, such as the IP Multimedia Subsystem (IMS). Testing this work in an environment like the IMS would provide more stimulating scalability results.
2. **Extension of TransferHTTP client to support Instant Messaging**
   Although SIP MESSAGE method is used in this implementation to transfer the Web session data, the implementation can still support Instant Messaging (IM) between two UACs. To extend our proposal to support IM, messages that are not wrapped in the XML format [1] for session transfer should be made to appear in a chat box UI. While this work shows interaction between Web browsers running on PCs, the work could also be extended to Web browsers on mobile devices. The stream media to call service, which is based on a P2P architecture, could also be extended to a client-server architecture (i.e. a broadcast service).
3. **Extension of TransferHTTP client to support to IPTV**
   In addition, our Web browser extension could be easily improved to support full multimedia services including IP Television (IPTV) [30, 67] and Internet TV. In general, it could be extended to support features available in the IMS clients. New plug-ins/codecs could be integrated into Web browsers in order to support file formats that are available in the IMS. For example, the video media type "video/3gpp" in the IMS has a file extension ".3gp" for video files.
   Moreover, this project could be extended to work with other IM services, such as GTALK, AIM and Yahoo Messenger. As earlier stated that an address book could be integrated into this project, the address book could be linked to users' address books from those IM services, and a suitable gateway, such as SIP-XMPP gateway, could be provided so that the Web browser extension (TransferHTTP) could work with those IM services.
4. **Extending TransferHTTP and CAS Interfaces to support buddy presence lists**
   An address book with presence of contacts could be integrated into the Web browser extension—TransferHTTP. Alternatively, CAS could be extended to provide the buddy list with presence. The integration of an address book or buddy list with presence would make it easier to retrieve the destination SIP

address and could make it possible to send requests to two or more people at a time.

5. **Smart Home Services via a SIP-based Web Browser**
The evolution of both mobile and home networking technologies offers opportunities for supporting the interoperation between mobile devices and devices residing in a home network. The transition to an all-IP wireless infrastructure in the mobile domain means that SIP is likely to be embedded into the next-generation mobile devices. Home networking technologies, such as Jini, UPnP, and HAVi, and protocols, such as Firewire, Bluetooth, X10, and IEEE 802.11, are proliferating the home networking market. OSGi [35] is one of the frameworks being defined to help manage the diversity and heterogeneity inherent in home networks. It is a Java-based framework that supports the delivery, activation and execution of services (called bundles) to home networks. It is independent of lower-level communication protocols and provides a middleware layer that can accommodate a variety of networking technologies [10]. Numerous works have already investigated the interoperability between SIP and the OSGi framework [10, 11, 68]. Although most works used a SIP-based device to control the smart home services, using a SIP-based Web browser to control the services is very practicable. While the SIP endpoint in the Web browser works at the control plane, the HTTP endpoint could help display the schematic diagram of the environment. A mashup service that renders, monitors, and controls smart home services via a SIP-based Web browser could also be explored.
6. **A SIP-based Web Services Model for Internet Telephony Services**
The Web services framework is built around SOAP. Although SOAP could act as a middleware to exchange information between peers in a decentralized and distributed environment, like SIP is used in this project, SIP outperforms SOAP in terms of routing messages, ensuring reliable delivery over an unreliable transport, ensuring security of message exchange and correlation of multiple SOAP messages [65].
An Internet telephony endpoint, such as a PDA or a 3G phone, is often resource restricted. It usually has low battery life, limited processing power and constrained storage space. Integrating a Web services framework into the resident SIP stack on such devices could be challenging. However, an XML parser and a small footprint SIP stack, which are loosely or tightly coupled, could enable an Internet telephony endpoint to participate in the Web services model. With numerous works exploring ubiqitious access to services [9, 17, 44] and convergence between Internet telephony services and Web services [24, 28, 41], this model could be explored.
7. **Improving the HTTP Session Mobility Service**
There was a problem encountered during the test of the HTTP session mobility service. The exact Web browser state could not be reproduced when session handoff was carried out on Mashups, AJAX-based and FRAME/IFRAME-based Websites. A probable solution to this problem is to capture the Web browser running state at the source UAC and reproduce at the destination UAC. While a session-based cookie expires in a short time of inactivity, a persistent cookie could provide access to a Website over a long period. An improved session management mechanism could be integrated into CAS so that a session handoff request could be held for a long time without expiring when the

destination SIP address cannot be reached or a Web server uses a session-based cookie. This feature could ensure that all session handoff requests do not expire for a certain period specified by the user.

### 6.2 Academic research efforts

Three primary academic research directions are currently under investigation.

1. **QoS-Enabled Architectural Scheme**
   A work that explored latency in browser-to-browser interaction is [27]. Latency in browser-based user-to-user interactions in our work was reported in [2]. A QoS-enabled SIP-based architectural scheme would be required in a bi-directional exchange of time-critical data between Web terminals. A QoS-Enabler, as shown in Figure 6, could be integrated into CAS to prioritize bandwidth consumption. In a multi-domain implementation (such as IMS), the QoS-Enabler in CAS will play the role of a gateway, thereby preprocessing QoS requests of applications and transforming them into requests to the underlying network QoS enforcement. Two or more CAS in the QoS-Enabled architectural scheme will act as a message overlay and enforce message policies for both text-based data (IM, URLs/Web session transfer) and multimedia data. They can accept configurations for QoS parameters, such as delay, jitter, packet loss and bandwidth, and send to the QoS-Enabler module. The module can also work with existing QoS entities in the IMS implementation. The configurations can be provided by application providers via the TransferHTTP API from a Web client side or via the CAS Web interface from a proxy side. We propose and discuss the QoS_Enabler in CAS as follows:
   CAS can only serve a limited number of concurrent requests. The QoS Module in Figure 6 implements additional control mechanisms that can provide different priorities to different requests. In this case, it is used to ensure that important

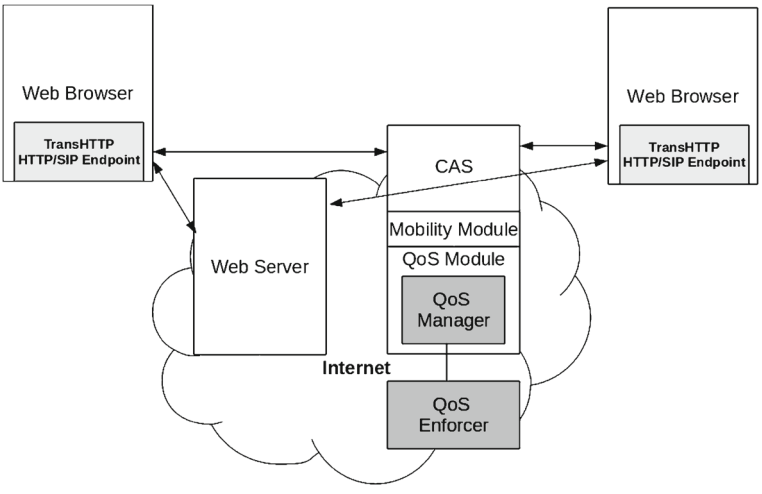


**Figure 6** CAS with a QoS-enabler

resources stay available under high server load. While the mobility module is used to accept, redirect or reject session transfer or call requests, the QoS module is used to control access to resources. That is, the QoS module can reject requests to unimportant resources or disable requests to very important resources while grant access to more important services for very important users. The key jobs of the QoS module are:

(a) Request level control: It controls the number of concurrent requests to CAS. It is used to define different priorities to different classes of service (messaging, session transfer and audio/video call) supported by CAS.
(b) Connection level control: It controls the number of TCP connections to CAS. This helps limit the connections coming from the TransferHTTP Web clients in order to reduce the maximum number of concurrent connections.
(c) Bandwidth level control: It throttles requests/responses to certain SIP addresses of TransferHTTP clients connected to CAS.
(d) Generic request line and header filter: It drops suspicious requests that can be a threat to CAS.

The QoS manager stores the QoS parameters for each user in a domain while the QoS enforcer applies the QoS policies to each request/response from a user. The QoS-related information include available classes of service, bandwidth allocated per class of service, key performance parameters per class (one-way packet delay, packet loss rate), call level quality parameters and control plane parameters. The XACML primary focus is access control and is supported by Mobicents. Hence, it can be used in implementing the QoS module. However, using the XACML policy implementation by Mobicents [62] in CAS would increase its complexity. Hence, we propose a new XML format as shown in Figure 7 for policy setting. A service provider can set the QoS for all users in a domain via the CAS Web interface. Alternatively, the policy file can be downloaded or uploaded to CAS. This approach will make it easier to set similar policies over multiple CAS in a multi-domain implementation. The QoS implementation would require extending the current XML message format from TransferHTTP client (as shown in [1] ) to support some additonal tags, such as priority level.

2. **Dynamic Composition of IMS Co-operative RIAs**
IMS Co-operative RIAs are RIAs that have IMS client functions integrated into them. The RIAs can provide richer Web services that leverage IMS services, such as call-control, media-control and QoS control that are currently unavailable on the Internet, to existing Web terminals. Nishimura et al. [34] proposed two architectures: (i) one that dynamically deploys a SIP/RTP component into a Web terminal for the consumption of IMS Co-operative RIAs, and (ii) another architecture that deploys the SIP/RTP component at an external gateway. For the former, the SIP/RTP component is also implemented as an RIA (SIP/RTP RIA) and dynamically deployed to a terminal at the same time the cooperative RIA is downloaded to the terminal. Once SIP/RTP is downloaded and run on a Web terminal, it can be cached or remain running on the terminal to reduce the dynamic deployment time. In the latter, the SIP/RTP component is terminated at an external gateway and a cooperative RIA calls APIs of the SIP/RTP component by a Web protocol, such as HTTP. As many non-PC terminals, such

**Figure 7** QoS XML message format

```
<?xml version="l.O" encoding="UTF–8"?>
<qos_params>
<sipdomain>cput.ac.za</sipdomain>
<qos_cos_list>
<cos status="accepted">call</cos>
<cos status="accepted">messaging</cos>
<cos status="accepted">session_transfer</cos>
</qos_cos_list>
<qos_request_level>
<request limit="minimum">8000</request>
<request limit="maximum">16000</request>
</qos_request_level>
<qos_connection_level>
<connection limit="minimum">32000</connection>
<connection limit="maximum">64000</connection>
</qos_connection_level>
<qos_bandwidth_level>
<sipuser id="sip:kbaolu001@cput.ac.za">
<bandwidth>100</bandwidth>
</sipuser>
</qos_bandwidth_level>
<qos_priority>
<priority status="accepted">normal</priority>
<priority status="accepted">non–urgent</priority>
<priority status="rejected">urgent</priority>
<priority status="rejected">emergency</priority>
</qos_priority>
</qos_params>
```

as TVs or game consoles, are not expected to be equipped with a microphone or camera, another terminal with input devices is necessary to provide interactive IMS co-operative RIAs. Extending TransferHTTP to dynamically consume IMS Co-operative RIAs like the first architecture proposed by [34] could be explored. In addition, a comparative study on the performance of both projects would be valuable to the converged services developer community.

3. **Effective Solutions for Large Deployments**
   Although most of the services in this project mirrored SIP session mobility signalling and GSM supplementary services (e.g. call blocking and call forwarding), significant changes were made to the signalling in order to develop a functional system. An example is when the destination SIP address cannot be reached, though it could accept a session transfer request from the source SIP address. In this case, the SIP proxy (CAS) will have to generate and send a 408 Timeout response to the source Web client; but a 408 Timeout is normally generated at a source UAC (User Agent Client) when its request cannot be processed within a specific time.
   Since CAS is designed to either block or forward a request, a request could however be picked-up later when the destination Web client registers with the proxy. In addition, the user can control ongoing session movement requests via CAS user interface (session tracking). The signalling however is slightly different from the usual SIP request-response signalling.
   Another issue was the implementation of partial blacklisting/whitelisting. In the real world, when a number is blacklisted by a user, it would not be able to send messages or make calls to that user. The situation is different here; CAS could

be configured in a way that it could block session transfer requests from a source Web client but allow a call set-up between the source Web client and its intended destination.

## 7 Conclusions

We claim that content sharing and proxy services in the support of the Web-browsing experience can relevantly encourage collaboration and community interaction among Internet users, working as a significant extension to the common Web infrastructure as the support for session handoff has recently demonstrated to do in mobility-enabled provisioning scenarios. Our research work, which was reported in some of our previous papers, demonstrates that the integration of a SIP stack into a Web browser has a very limited negative effect on memory footprint and quality of experience; the result is relevant because it shows that the inclusion of SIP in commercial Web browsers is not only feasible, but also capable of easily offering new value-added services to end users. This article contributes to the state-of-the-art in the field by comparing our implemented prototypes with emerging industry works, such as WebRTC, Google Wave, and Open APIs, thus better positioning our proposal and better pointing out its original technical elements. In addition, we claim that there are several directions of application/extension of the research work done, of both academic and industrial interest; these directions are part of our ongoing research activities and may be research opportunities of relevant interest for the community in the field given the open-source availability of the developed tools. Moreover, these directions show the vast applicability of the proposed tools and their potential relevance in many application domains and deployment scenarios, as a valuable support towards the ambitious goal of the wide-scale adoption of fully converged applications and services.



## References

1. Adeyeye, M., Ventura, N.: A SIP-based Web client for HTTP session mobility and multimedia services. In: The Special Section on Hot Topics in Mesh Networking. The International Journal for the Computer and Telecommunications Industry. Elsevier Computer Communications (COMCOM), vol. 33, issue 8, pp. 954–964 (2010). doi:10.1016/j.comcom.2010.01.015
2. Adeyeye, M., Ventura, N., Foschini, L.: Converged multimedia services in emerging Web 2.0 session mobility scenarios. Wirel. Netw. **18**(2), 185–197 (2012). doi:10.1007/s11276-011-0394-z
3. Adeyeye, M., Ventura, N., Humphrey, D.: Control services for the HTTP session mobility service. In: Proceedings of the 3rd IEEE Conference on New Technologies, Mobility and Security (NTMS), 21–23 Dec 2009. Cairo, Egypt (2009)
4. A prototype of Mozilla Weave: http://labs.mozilla.com/2007/12/introducing-weave/. Accessed 10 June 2008
5. Bhat, R.R., Tait, D.: JAVA APIs for integrated networks. In: Jespen, T. (ed.) Java in Telecommunications: Solutions for Next Generation Networks, p. 193. Wiley & Sons, New York (2001)
6. Bond, G., Cheung, E., Fikouras, I., Cheung, E., Levenshteyn, R.: Unified telecom and Web services composition: problem definition and future directions. In: Proceedings of the IPTCOMM 2009, 7–8 July 2009. Atlanta, Georgia (2009)

7. Boswell, D., King, B., Oeschger, I., Collins, P., Murphy, E.: Creating Applications with Mozilla, 1st edn, pp. 1–8. O'Reilly Press, USA (2002)
8. Canfora, G., Di Santo, G., Venturi, G., Zimeo, E., Zito, M.V.: Proxy-based handoff of Web sessions for user mobility. In: Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous '05) (2005)
9. Cesar, P., Vaishnavi, I., Kernchen, R., Meissner, S., Hesselman, C., Spedalieri, A., Boussard, M., Bulterman, D.C.A., Gao, B.: Multimedia adaptation in ubiquitous environments: benefits of structured multimedia documents. In: Proceedings of the 8th ACM Symposium on Document Engineering (DocEng '08), 16–19 Sept 2008, pp. 275–284. Sao Paulo, Brazil (2008)
10. Chang, G., Zhu, C., Ma, M.Y., Zhu, W., Zhu, J.:Implementing a SIP-based device communication middleware for OSGi framework with extension to wireless networks. In: Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS '06), 20–24 June 2006, pp. 603–310. Zhejiang, China (2006)
11. Chintada, S., Sethuramalingam, P., Goffin, G.: Converged services for home using a SIP/UPnP software bridge solution. In: Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC '08), 10–12 Jan 2008, pp. 790–794. Las Vegas, USA (2008)
12. Chou, W., Shan, X., Li, J.J.: An architecture of wireless Web and dialogue system convergence for multimodal service interaction over converged networks. In: Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC '03), 30 Sept.–3 Oct. 2003, pp. 69–74. Hong Kong (2003)
13. Corradi, A., Lodolo, E., Monti, S., Pasini, S.: Dynamic reconfiguration of middleware for ubiquitous computing. In: Proceedings of the Third Workshop on Adaptive and DependAble Mobile Ubiquitous Systems, ADAMUS 2009, 13–17 July 2009, pp. 7–12. London, United Kingdom (2009)
14. Deruelle, J.: JSLEE and SIP-Servlets interoperability with mobicents communication platform. In: Proceedings of the 2nd International Conference on Next Generation Mobile Applications, Services and Technologies, 16–19 Sept 2008. Cardiff, Wales, United Kingdom (2008)
15. Google Browser Sync: http://www.google.com/tools/firefox/browsersync/. Accessed 10 June 2008
16. Gurbani, V.K., Sun, X.-H.: Terminating telephony services on the internet. IEEE/ACM Trans. Netw. **12**(4), 571–581 (2004)
17. Gurbani, V.K., Sun, X.-H., Brusilovsky, A.: Inhibitors for ubiquitous deployment of services in the next-generation network. IEEE Commun. Mag. **43**(9), 116–121 (2005)
18. Handley, M., Jacobson, V.: SDP: Session Description Protocol. IETF RFC 2327. Accessed 12 Apr 2012
19. Handley, M., Schulzrinne, H., Schooler, E., Rosenberg, J.: SIP: Session Initiation Protocol. IETF RFC 2543 (1999)
20. Hsieh, M.-D., Wang, T.-P., Tsai, C.-S., Tseng, C.-C.: Stateful Session Handoff for Mobile WWW, vol. 176, pp. 1241–1265. Information Sciences, Elsevier Science Press (2006)
21. IETF RTCWeb-SIP WG: http://tools.ietf.org/html/draft-kaplan-rtcweb-sip-interworking-requirements-01. Accessed 20 Nov 2011
22. IETF WebRTC: http://tools.ietf.org/wg/rtcweb. Accessed 20 Nov 2011
23. Jain, R., Bakker, J., Anjum, F.: Programming Converged Networks: Call Control in Java, XML, and Parlay/OSA, 1st edn, pp. 27–34. Wiley Interscience, Canada (2005)
24. Kataoka, H., Toyama, M., Sueda, Y., Mizuno, O., Takahashi, K.: Demonstration of Web contents collaborative system for call parties. In: Proceedings of the 7th IEEE Consumer Communications and Networking Conference (IEEE CCNC '10), 9–12 Jan 2010. Las Vegas, Nevada, USA (2010)
25. Krechmer, K.: Open standards: a call for change. IEEE Commun. Mag. **47**(5), 88–94 (2009)
26. Larson, J.: VoiceXML and the W3C speech interface framework. IEEE Multimedia **10**(4), 91–93 (2003)
27. Linner, D., Stein, H., Staiger, U., Steglich, S.: Real-time communication enabler for Web 2.0 applications. In: Proceedings of the Sixth International Conference on Networking and Services (ICNS '10), 7–13 Mar 2010, pp. 42–48. Cancun, Mexico (2010)
28. Maes, S.H.: A call control driven MVC programming model for mixing Web and call or multimedia applications. In: Proceedings of 4th International Conference on Mobile Technology, Applications and Systems (Mobility 2007), 10–12 Sept 2007. Singapore (2007)
29. Manzalini, A.: Tomorrow's open internet for telco and Web federation. In: Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems (CISIS '08), 4–7 Mar 2008, pp. 567–572. Barcelona, Spain (2008)

30. Mas, I., Berggren, V.: IMS-TV: an IMS-based architecture for interactive, personalized IPTV. IEEE Commun. Mag. **46**(11), 156–163 (2008)
31. Mate, S., Chandra, U., Curcio, I.D.D.: Moveble-multimedia: session mobility in ubiquitous computing ecosystem. In: Proceedings of the Mobile and Ubiquitous Multimedia (MUM '06). Stanford, California (2006)
32. Mulligan, C.E.A.: Open API standardization for the NGN platform. IEEE Commun. Mag. **47**(5), 108–113 (2009)
33. Munkongpitakkun, W., Kamolphiwong, S., Sae-Wong, S.: Enhanced Web session mobility based on SIP. In: Proceedings of the 4th International Conference on Mobile Technology, Applications and Systems (Mobility 2007), 10–12 Sept 2007, pp. 346–350. Singapore (2007)
34. Nishimura, H., Ohnishi, H., Hirano, M.: Architecture for Web-IMS co-operative services for Web terminals. In: Proceedings of the 13th International Conference on Intelligence in Next Generation Networks (ICIN '09), 26–29 Oct 2009, pp. 1–6. Bordeaux, France (2009)
35. Open Service Gateway Interface Specification: Release 4. http://www.osgi.org. Accessed 24 March 2010
36. Rosenberg, J.: A Presence Event Package for the Session Initiation Protocol (SIP). IETF RFC 3856 (2004)
37. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol. IETF RFC 3261 (2002)
38. Saint-Andre, P. (ed.): Extensible Messaging and Presence Protocol (XMPP): Core. IETF RFC 3920 (2004)
39. Sarin, A.: The future of convergence in the communications industry. In: The Technology Leaders Forum, IEEE Communications Magazine, pp. 12,14 (2007)
40. Saxtoft, C.: Convergence: User Expectations, Communications Enablers and Business Opportunities, pp. 34–37. John Wiley & Sons Ltd, England (2008)
41. Sbata, K., Khrouf, H., Zander, S., Becker, M.: Converging Web and IMS services: stakes and solution proposals. In: Proceedings of the International ACM Conference on Management of Emergent Digital EcoSystems (MEDES '09), 27–30 Oct 2009. Lyon, France (2009)
42. Schonwalder, J., Fouquet, M., Rodosek, G.D.: Future Internet = Content + Services + Management. IEEE Commun. Mag. **47**(5), 27–33 (2009)
43. Schulzrinne, H.: The tel URI for Telephone Numbers. IETF RFC 3966 (2004)
44. Shacham, R., Schulzrinne, H.: Ubiquitous device personalization and use: the next generation of IP multimedia communications. In: The Journal of ACM Transactions on Multimedia Computing, Communications and Applications, vol. 3, issue 2, article 12 (2007)
45. Shacham, R., Schulzrinne, H., Thakolsri, S., Kellerer, W.: Session Initiation Protocol (SIP) Session Mobility. Internet-Draft: draft-shacham-sipping-session-mobility-05. ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-shacham-sipping-session-mobility-05.txt. Accessed 18 Nov 2007
46. Silvana, G.P., Schulzrinne, H.: SIP and 802.21 for service mobility and pro-active authentication. In: Proceedings of the Communication Networks and Services Research Conference (CNSR '08), pp. 176–182. Halifax, Nova Scotia, Canada, (2008)
47. Song, H.: Browser session preservation and migration. In: Poster Session of WWW 2002, 7–11 May, p. 2. Hawai, USA (2002)
48. Sun Microsystems: JAIN SIP Release 1.0 Specification (2001). http://www.jcp.org/aboutJava/communityprocess/final/jsr032. Accessed 25 March 2010
49. Sun Microsystems: JAIN SIP Release 1.1 Specification (2001). http://www.jcp.org/en/jsr/detail?id=289
50. The Akogrimo Project: http://www.mobilegrids.org. Accessed 19 June 2008
51. The Google Wave Click-to-dial: http://googlewavedev.blogspot.com/2009/06/twiliobot-bringing-phone-conversations.html. Accessed 31 July 2009
52. The Google Wave Project: http://wave.google.org. Accessed 5 July 2009
53. The Mobicents Open Source SLEE and SIP Server: http://www.mobicents.org/index.html. Accessed 20 Jan 2009
54. The Mozilla Firefox Web browser: http://www.mozilla.org. Accessed 12 Apr 2012
55. The OmniVox3D Application Server: http://www.apexvoice.com/index.php/158/sip-application-server. Accessed 1 Sept 2008
56. The Opera Multimodal Browser: http://www.opera.com/products/devices/multimodal/. Accessed 1 Sept 2008
57. The PJSIP Project: http://www.pjsip.org. Accessed 12 Apr 2012
58. The TransferHTTP Controller: http://transferhttp.berlios.de. Accessed 8 July 2009
59. The TransferHTTP Extension: http://transferhttp.mozdev.org. Accessed 8 July 2009

60. The Voxeo Prophecy Platform: http://www.voxeo.com/prophecy. Accessed 1 Sept 2008
61. The Wave Protocol: http://www.waveprotocol.org. Accessed 5 July 2009
62. The XACML Policy Implementation in Mobicents: http://www.jboss.org/picketlink/XACML. Accessed 17 Jan 2012
63. WebRTC: http://www.webrtc.org. Accessed 20 Nov 2011
64. Web Sharing and RichDraw: http://hyperstruct.net/2007/2/24/xml-sync-islands-let-the-Web-sharing-begin. Accessed 4 Sept 2009
65. World Wide Web Consortium (W3C): SOAP Version 1.2 Part 0: Primer (Second Edition). Technical Recommendation. http://www.w3.org/TR/soap12-part0/. Accessed 1 Apr 2010
66. Wu, X., Schulzrinne, H.: Use SIP MESSAGE method for shared Web browsing. http://www3.tools.ietf.org/id/draft-wu-sipping-Webshare-00.txt. Accessed 14 Nov 2001
67. Xiao, Y., Du, X., Zhang, J., Hu, F., Guizani, S.: Internet protocol television (IPTV): the killer application for the next-generation internet. IEEE Commun. Mag. **45**(11), 126–134 (2007)
68. Yang, J., Park, H.: A design of open service access gateway for converged Web service. In: Proceedings of the 10th International Conference on Advanced Communication Technology (ICACT '08), 17–20 Feb 2008, pp. 1807–1810. Gangwon-Do, South Korea (2008)