Websocket Enabler: achieving IMS and Web services end-to-end convergence

Gerard Nicolas
Orange Labs
38, rue du general Leclerc
Issy-les-Moulineaux, 92794 France
gerard1.nicolas@orange-ftgroup.com

Karim Sbata
Orange Labs
38, rue du general Leclerc
Issy-les-Moulineaux, 92794 France
karim.sbata@orange-ftgroup.com

Elie Najm
Télécom ParisTech
46, rue Barrault
Paris, 75013 France
elie.najm@telecom-paristech.fr

ABSTRACT
Over the last few years, significant evolutions such as the mobile phones' enhanced web-browsing capabilities and the technical incursion of Web major players into the Telco world (e.g. Google, Facebook) have reduced the gap between Telecom and Web worlds. In this context, converging IMS and Web service platforms has become a key challenge that needs to be addressed by both Web and Telecom players. Several interesting solutions, illustrating different convergence approaches, have been proposed so far. Unfortunately, none of them has been able to provide an efficient way to set up end-to-end converging services. Indeed, Web-based applications are synchronous, as they rely on HTTP. On the other hand, IMS services can be provided in both asynchronous and synchronous modes. We define synchronous applications as services in which each provided resource or piece of information has to be explicitly requested by the consumer and asynchronous applications as services that can notify their consumers anytime they need. But recently, the W3C and the IETF have released new standards (HTML5 and Websocket protocol), introducing important evolutions in the Web paradigm. In particular, the Websocket technology allows a native support for asynchronous Web applications. Our proposal is a converging framework (called WSE, standing for Websocket Enabler) that takes advantage of this new technology to achieve end-to-end service convergence.

Keywords
Web-Telco convergence, Websocket, IMS, SIP.

1. INTRODUCTION
Information Technology becomes pervasive in our everyday's life, in both our professional and personal activities. Communication devices are widely spread and used by end-users to access a growing number of services. These services are provided by Web players through the Internet or by Telco operators through the IP Multimedia Subsystem (IMS). Each of these environments relies on specific protocols and architectures that are not natively interoperable. Indeed, the IMS is a SIP-based architecture that provides real-time and asynchronous services whereas the Internet is HTTP-based and was initially conceived to provide synchronous client-server services. But Telco operators want to take advantage of the Web openness in order to offer their services to a higher number of users and Internet service providers are interested by operator's assets which are security, quality of service and reliable transport infrastructure. This creates the need for studying how these two worlds can be combined to offer new services which benefit from their convergence.

Several studies tried to find a means to achieve this convergence, either by enhancing end-user's device application layer, or within the network application layer or within the network infrastructure. The most significant solutions that have been proposed rely on protocol conversion gateways or client-based integrations (e.g. Flash soft-phones for Web browsers). But no standard solution has been defined yet, and all proposed approaches are proprietary to their implementers. This raises the need to have a novel solution relying on open standards. This study proposes a new convergent architecture permitting unified protocol agnostic access to Web-based and IMS-based services.

The following section (section 2) describes the technical and functional aspects of our proposal and highlights its interest to all business players involved in service provisioning. Section 3 analyses our approach and points out its pros and cons.

Conference'10, Month 1–2, 2010, City, State, Country.

2. OUR CONTRIBUTION
In this section, we describe our proposed solution that achieves convergence between Web and IMS worlds. In subsection 2.1 we detail the architecture of the proposal whereas subsection 2.2 refers to an illustrative use case showing how this architecture can provide convergent services.

2.1 The proposed architecture
This subsection deals with the architectural aspect of our proposed solution for Web/Telco convergence. The central part of the solution is an entity called the Websocket[1] Enabler (WSE). We first describe the environment of the WSE, i.e., the entities that interact with it. Then we turn to the presentation of the internal structure and components of the WSE.

2.1.1 The WSE technical environment
WSE is the central part of the proposed solution. It is in interaction with an environment composed of: the end-user's Web browser, the ID proxy, the retailer's Web application and the service connectors.

2.1.1.1 Service connectors
The WSE is interfaced with service providers through service connectors. A service connector is a gateway that interfaces a service (or a set of services) with the WSE IN and OUT connectors (explained later). The main objective of these connectors is to facilitate the integration of back-end service providers.

2.1.1.2 End-user client
End-users connect to the WSE by accessing the retailer's site through Websocket-compliant Web browsers. Currently, most of the Web browsers are compliant (Microsoft Internet Explorer just

released a new HTML5 version which supports Websocket through a plug-in).

2.1.1.3 The retailer's Web application

The retailer's Web application is the entry point for the end-user. Through specific user journeys, it provides the end-user with the possibility to establish a WSE session.

2.1.1.4 Client JavaScript libraries

To interact with the WSE, the retailer should include a set of JavaScript libraries that allow for different client accesses to the retailer site. The first one is an authentication script provided by the ID proxy that allows the network operator to authenticate the end-user. It provides the Web browser with a user token that should be used to connect to the WSE. Once the user is authenticated by the operator, a set of JavaScript libraries are included in the Web page. The first one is the wse.js which is provided by WSE and has three functionalities: Websocket initialization and life-cycle, module loading and message handling (in/out). This library is mandatory regardless of the used services. In addition to this main library, each time a service is included to the WSE session, a corresponding library is loaded. For example, for SIP-based services like Presence, the browser loads the sip.js module. The Web application can then register the user to SIP or IMS servers and exchange SIP messages with them.

2.1.2 The WSE components

The WSE is defined as the operator's access interface to Internet domain at transport and application layers. It has several functionalities, like for instance handling Websocket connections to browsers. This is mainly done at transport layer. In addition it manages services logic at application layer and also connection to operator's data bases as well as interconnections with application servers where the value added services reside. This node groups five components as illustrated in Figure 1.

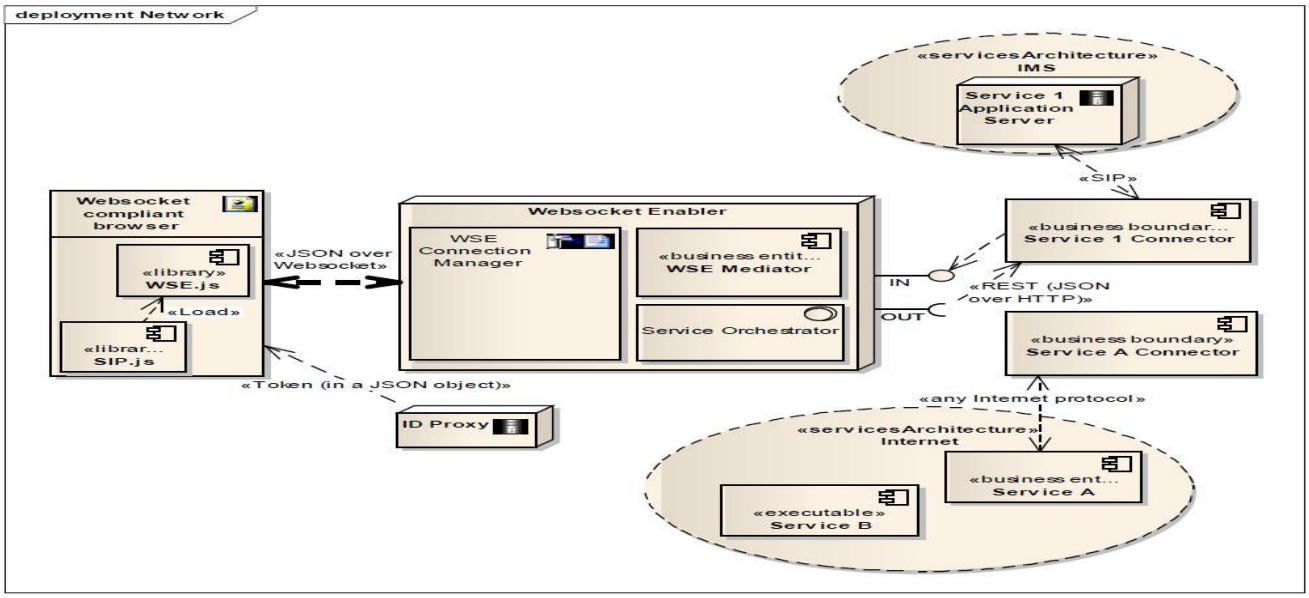

Figure 1. WSE architecture

2.1.2.1 The WSE connection manager

It is the server that communicates with browsers in order to establish Websocket connections. It maintains Websocket session and terminates it when it reaches timeout. In addition it supervises Web user authentication in operator's network. The HTML5 technology enhances the server's ability to manage a high number of Websocket connections with a minimal cost. This is because Websocket discharge the servers compared to HTTP where a large header must be analyzed to process a request from the client.

2.1.2.2 The WSE Mediator

Its main role is to manage message routing between the service provider and the client, routing which is based on service identity. For services that request access to operator's data bases, access rules are defined in this same component. Also, the operator or retailer can propose to its client service alternatives in cases where a specific service usage is overloaded.

2.1.2.3 The service orchestrator

Its role is to orchestrate services composition in order to provide a novel value added service as a response to end-users' need. We use Orc [6] as an orchestration language. As Orc is a high layer orchestration language that can manage synchronous and asynchronous services independently of their implementations, it allowed us to orchestrate sites where a site is defined as a single service (Web or Telecom) developed in any language.

The orchestrator is composed of three functional entities: the first is the Orc Service Definition (OSD) where services compositions are defined using Orc expressions. The second is the Composite Services Definitions Repository (CSDR) where defined Orc compositions (Orc code) are stored. The third is the Orchestration Engine (OE) which executes Orc scripts. Orchestration functionality can be hosted on any machine independently from Internet service providers and network operators. In our approach, we choose to make the orchestrator a functional entity of the WSE and hosted by the broker, as it manages also the identity of the end-user.

2.1.2.4 The connector IN

It is a REST/JSON [8] [9] interface between the cloud of service connectors and the consumers. It receives all services' data pushed to end-users as a response to a request (synchronous mode) or as soon as data is available (asynchronous mode). On

reception, the IN connector will forward the data to the orchestrator or to WSE mediator component depending on whether it is a composed or basic service. The network operator can use this component to control data load tunneled to Websocket Enabler and to have a more global idea of the percentage of composed services.

2.1.2.5 The connector OUT

Like the Connector IN, the Connector OUT is also a REST/JSON interface but its role is to receive clients' requests and data and send them to services. This component will aggregate data sent from WSE. In addition, it manages process IDs in order to instruct the IN connector to which process received data must be assigned. This enhances operator's control on data traffic.

2.2 Enhanced Address Book: an illustrative use case

An illustrative use case showing how our proposed architecture can provide converging services and orchestrate them efficiently is detailed in[3]. The main idea of this use case is to allow Web application providers to offer to their end-users an access to an enhanced address book through their Web sites. This enhanced address book consists of a list of contacts whose information cards can be enriched with location and presence information when available.

3. DISCUSSION AND OPEN ISSUES

Our approach for Web-IMS convergence has some advantages presented hereunder.

3.1 Websocket advantages for interactive Web applications

This concerns the Internet part of our architecture which is between Web clients or browsers and the WSE. In the context of interactive converging services (mainly asynchronous), the Websocket technology provides 500:1 even 1000:1 reduction of traffic overhead and 3:1 reduction in latency compared to HTTP-based solutions[4]. Moreover, the uniqueness of the Websocket connection during the service session and the reduction of the overhead will decrease the computing resources needed by the Web servers to process the exchanged application data. Finally, the latency reduction enhances the end-user experience by increasing its interactivity. All these advantages make Websocket a reliable transport means for the Web, to provide a more efficient Web-Telco converging services support.

3.2 Convergent orchestration

In the global context of service architectures, orchestration is a key functionality and the efficiency of its implementation impacts significantly the performance of the service platform. In the particular context of convergent services, these architectures should rely on an agnostic orchestration able to manage the composition of synchronous Web-oriented services with asynchronous Telco services (IMS or SIP–based). This is the reason why we define in our approach an orchestration entity based on a service agnostic orchestration language Orc. This entity is able to manage the composition of services regardless of their type.

3.3 Open issues

We have not yet covered in our approach all the problems that may arise when orchestrating SIP-based and Web-based services. A good assessment of these problems is given in [7]. On another issue, our approach is dependant on the Websocket standard which is known to have some security holes which are currently being addressed by the standardization bodies. Scalability need to be addressed as brokers are unique entry points for the converged flows and an appropriate load balancing solution should be devised. The commercial success of the proposed architecture is also dependant on the establishment of appropriate partnerships, between the *retailers*, the *brokers* and the *third party providers*.

4. CONCLUSION AND PROSPECTS

We propose a convergent architecture based on Websocket that provides access to Web and telecom services seamlessly. In our proposal, the Websocket technology ensures a southbound or core network convergence. In addition, the agnostic orchestration functionality based on service connectors and the service logic components ensures application layer convergence. Device application layer convergence is guaranteed by JavaScript libraries loaded into the browser (wse.js and sip.js modules). Hence we achieved Web-Telco southbound, northbound and device application layer convergence.

After defining the overall architecture, our next objective is to define an efficient agnostic orchestration framework processing synchronous and asynchronous services equally. To this end we undertook a deep study of the Orc orchestration language semantics and we began to design and develop a composite service that will be orchestrated by Orc. This allows assessing this technology. Concerning the ongoing standardization effort on the HTML5, we support the enhancement of Websocket to encompass real-time audio and video in full-duplex mode [2]. This added feature will allow further integration of multimedia services and their delivery through web browsers.

5. ACKNOWLEDGMENTS

This work is supported by SERVERY[5] an European project of the Celtic initiative, whose goal is to enable a service Marketplace that bridges the Internet and Telco worlds.

6. REFERENCES

[1] The Websocket protocol, February 25, 2011, http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-06

[2] RTC Web Workshop, October 6, 2010, http://rtc-Web.alvestrand.com

[3] Enhanced Address Book Use Case, http://personalapis-test.orange.fr/eab/

[4] Lubbers, P., Greco, F., 'HTML5 Websockets: A Quantum Leap in Scalability for the Web'. Http://Websocket.org/quantum.html, Kaazing Corporation.

[5] Fodor, S., 2008-2011, Advanced Service Architecture and Service Delivery Environment, http://projects.celticinitiative.org/servery/

[6] Orc Language Project, http://orc.csres.utexas.edu/

[7] Bond G; Cheung E; Fikouras I; Levenshteyn R. Unified Telecom and Web Services Composition: Defining the Problem and Future Directions. IPTCOMM'09. Atlanta, July 2009. LNCS - Springer Verlag.

[8] JSON specification, http://www.json.org/

[9] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm