Architecting end-to-end convergence of Web and Telco services

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

Karim Sbata
Orange Labs
38, rue du general Leclerc
Issy-les-Moulineaux, 92794 France
karim.sbata@orange.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 or Internet Protocol Multimedia Subsystem 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) presents several significant Web-IMS convergence solutions that have been proposed over the last few years. Section 3 describes the technical and functional aspects of our proposal and highlights its interest to all business players involved in service provisioning. Section 4 analyses our approach and points out its pros and cons.

2. RELATED WORK
Web and IMS convergence has been a hot topic during this last five years. Several telecommunication actors have proposed convergent solutions based on different approaches. These approaches can be classified in three different categories: device convergence, achieved by the user application layer; middleware convergence, achieved by the back-end application layer; core convergence, achieved by core network nodes. This classification is used in the following subsections. In addition service orchestration has been an important criteria to determine interoperability between Web and Telco services.

2.1 Device Convergence
Convergence at device application layer aims to find one protocol unifying access to all types of services. Two propositions are presented hereunder.

Family Portal [4]: is an application that allows voice and video calls between family members. It integrates Web 2.0 technologies with IMS. This solution includes an innovative device called IMS Smartdongle which allows a user to make phone call from any PC by inserting the device into a USB port. This USB device contains a SIM application, an authentication applet, a browser and a SIP soft-phone. Thus calls can be made via a Web page in a browser launched on a mobile phone.

Flash solution [1]: to deliver personal communication services on IMS or Web infrastructure two solutions where proposed based on Flash proprietary technology. The first is client based where Web client must be aware of callee's IMS SIP URI and must be able to play the role of a User Agent (UA). The second is based on a Back-To-Back User Agent (B2BUA) that establishes a Flash connection with the Web user and a SIP connection with the IMS callee. These two solutions must manage signaling conversion from any proprietary protocol on the Internet to SIP on IMS network, as well as stream transcoding and Real-Time Messaging Protocol (RTMP) to Real-Time Transport Protocol (RTP)/Real-Time Transport Control Protocol (RTCP) conversion.

2.2 Application Convergence

This is known as the northbound convergence. Two initiatives deal with the network application layer convergence: WIMS 2.0 and IMS 2.0.

WIMS 2.0: in this initiative, the IMS network is considered as the base platform to construct convergence between Telecom and Web 2.0 by exposing IMS capabilities through open Web Application Programming Interfaces (APIs) following the RESTful[16] approach. IMS sessions are modeled as resources which are represented by URIs. As an example IMS session is accessed by Web 2.0 through *OpenAPIsRoot / IMPU / Service / SessionID*. It represents a session resource identification following REST style of architecture [3].

IMS 2.0: this architecture enables the discovery and better control of IMS and Web 2.0 resources. It is based on three network entities: the service broker included in IMS network, it is responsible for registering, orchestrating and hosting common IMS 2.0 resources. The Web 2.0 gateway (logical interface) sits at the border between IMS and Web 2.0 world. It acts as an HTTP/SIP protocol converter and exposes IMS assets to Web 2.0 as REST URIs. Finally the resource database holds service information, user types, permissions and user profile. By combining IMS, Web 2.0 and third party service providers, a user will require only one subscription to IMS world in order to benefit from all services offered by those three actors[2].

2.3 Network Convergence

This is known as the southbound convergence and it puts IMS/Web convergence at the core network level. An illustration of this type of convergence is the WSC architecture (stands for Web Session Controller)[5]. It is a border entity between IMS and the Web. The goal of this solution is to merge IMS and Web sessions within the same hybrid application guaranteeing to the operator the control of the service. WSC manages interactions with partner's Web application servers and has two functional elements. The first is the Web Control Function (WCF) which performs adaptation between IMS signaling and the Web. The second is the Web Media Gateway (WMG) which manages interworking between the two worlds. WSC offers to Web service providers the possibility to monetize their services via IMS network charging functionalities and allows blending Web content delivery with IMS services such as presence, instant messaging and multimedia telephony applications.

2.4 Service Orchestration

Several service orchestration languages exist today, however the need to orchestrate Web and non-Web (Telco) services is not handled by the most famous one WS-BPEL. For this purpose we studied a high level orchestration language (Orc) which orchestrates sites (services) implemented in any language independently of their interfaces. Orc [12] uses simple concurrency primitives. It is available for casual users who want to try Orc in the browser, for Eclipse users who want to develop Orc programs on their machine and for Command line users (using orc.jar). So this language is implemented and runs now on its 2.0.2 version.

3. OUR CONTRIBUTION

In this section, we describe our proposed solution that achieves convergence between Web and IMS worlds by working on all of the three convergence places thus performing core network, device and network application layer convergence in the same framework. In the following subsections (subsections 3.1 and 3.2) we describe the business players and the involved technologies of our proposal; then, in subsection 3.3 we detail the architecture of the proposal. Finally subsection 3.4 presents a use case call flow.

3.1 Business Players

As known, any technical work has a business goal and is conducted in the context of a business model. Our proposal is about an offer of services involving multiple players pursuing different business goals. The players are: the consumer, the retailer, the broker, the connectivity and third party provider. Figure 1 illustrates business relationships between these players.

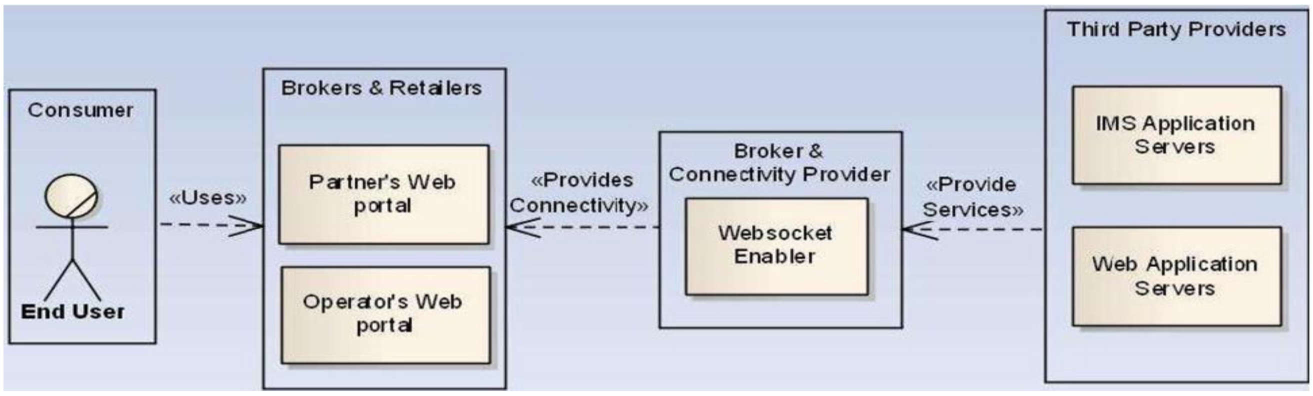

Figure 1. Business players

The *consumer* of a service is the end user who has access to the service and directly benefits from its delivery. He/she can be the service subscriber, or act in the context of a subscription by some other entity (an individual, a household, or a company). Depending on the provided service, the *consumer* can also act without the context of any subscription.

The *retailer* is the entity that provides the service directly to the *consumer*. It can be the *broker* or one of its partners.

The *broker* is the entity that provides a single entry point to *consumers,* allowing them to access IMS, Web-Based, or orchestrated services. It has also the usual broker's role which is to provide stakeholders with information that enables them to find other stakeholders and services. The broker in our system can be a Telco or an Internet player.

The *third party provider* provides service logic or/and content to retailers or other third party providers. Offered services can originate from either IMS Application Servers, or Web Application Servers, or can be orchestrations of the above (performed by the orchestrator residing in the Websocket Enabler introduced in subsection 3.3). Thus, telecom and Internet players are also potential third party providers.

Content is transported by a physical network which is the union of both operator and World Wide Web networks. Thus IMS and Web infrastructures play the role of *connectivity providers*.

3.2 Involved Technologies

This section describes the technological context of our solution. Technologies involved in this work are divided into two categories: client-access technologies and service back-end technologies.

The technologies involved on client-side are mainly Websocket protocol which enables full-duplex communication between a user agent running untrusted code in a controlled environment and a remote host[6] and HTML5. The latter is a new version of browser's Markup language which is characterized by new media tags, a better organization of the code itself and some other improvements. But the most important add-ons are the number of JavaScript APIs that help Web application creation and mainly the *Websocket API.* It plays the role of an interface between the browser that runs HTML5 and Internet transport protocols.
Back-end technologies are mainly IMS services and Web service platforms. They are provided through the IMS network or third party Internet players. The IP Multimedia Subsystem (IMS) is an architectural framework for delivering Internet Protocol (IP) multimedia services. It was originally designed by the wireless standards body 3GPP as a part of the vision for evolving mobile networks beyond GSM. The IMS core is divided into 3 main layers: common IP core, common service capabilities and service layer. It is characterized by the usage of the SIP protocol for signaling and control functions.

Back-end services can be provided also by Web service platforms Web back-end services are synchronous and can be REST or SOAP[14] based. REST architectural style represents the Web as a group of resources (services) identified by URIs and accessed by HTTP predefined methods. These services are lightweight and easy to build. SOAP based Web services are known as Service Oriented Architecture (SOA) components which are widely deployed in enterprise computer networks. SOAP Web services are rigid, allowing to type check the adherence of services to pre-established contracts.

3.3 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 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.

3.3.1 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.

3.3.1.1 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.

3.3.1.2 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).

3.3.1.3 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.

3.3.1.4 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.

3.3.2 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 2.

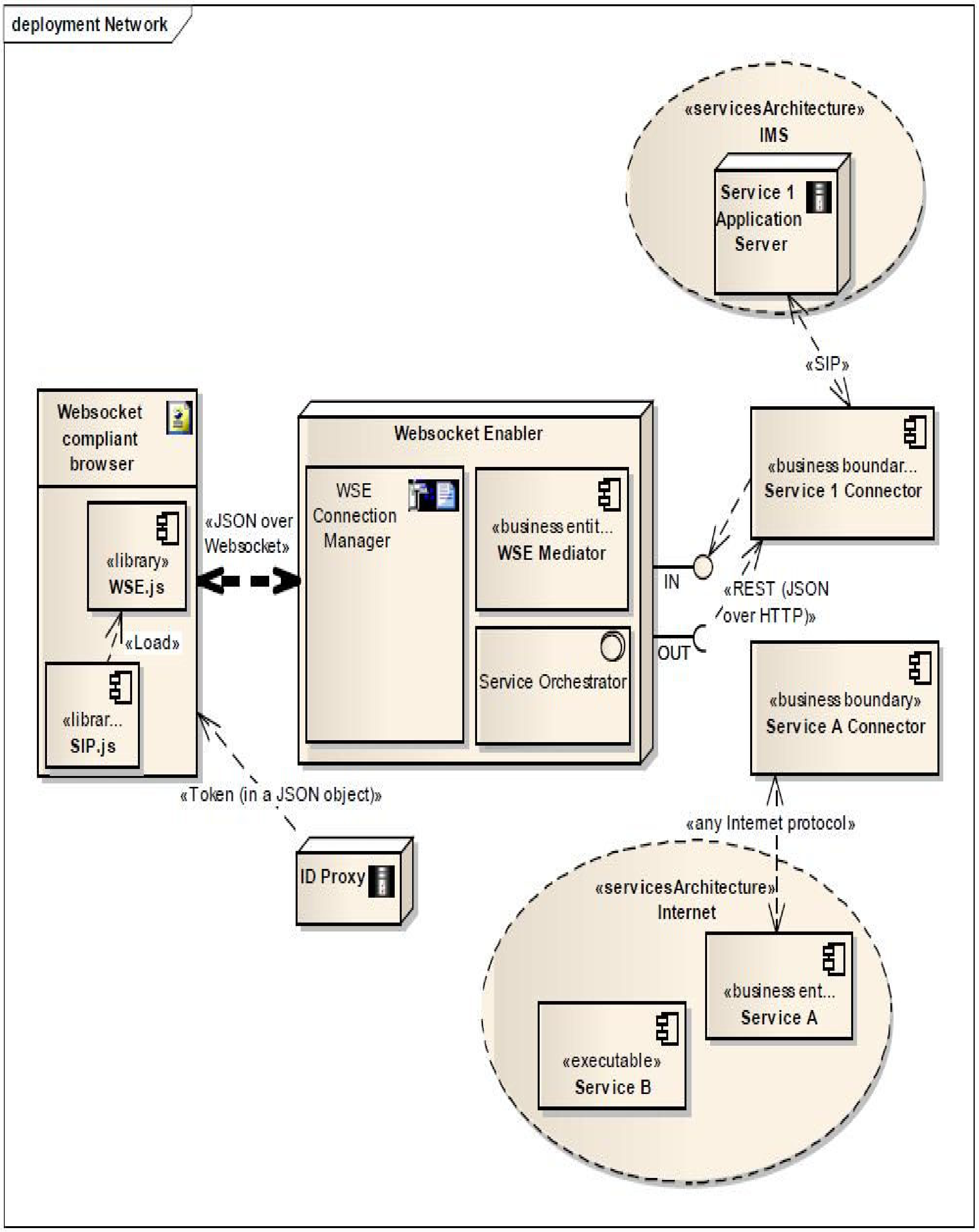

Figure 2. WSE architecture

3.3.2.1 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.

3.3.2.2 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.

3.3.2.3 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 [12] 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.

3.3.2.4 The connector IN
It is a REST/JSON[15] 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.

3.3.2.5 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.

3.4 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 [8]. 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. Figure 3 and Figure 4 show the call flow of the use case.

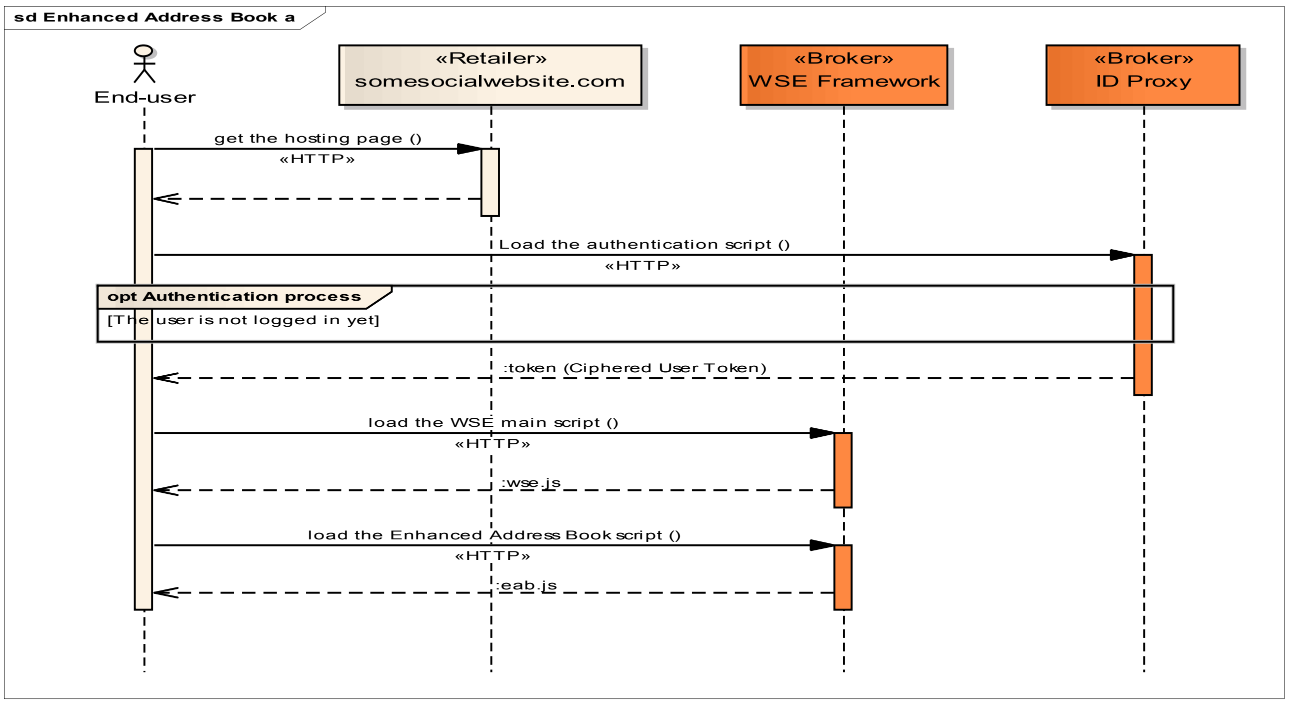
sd Enhanced Address Book a
End-user
«Retailer» somesocialwebsite.com
«Broker» WSE Framework
«Broker» ID Proxy
get the hosting page ()
«HTTP»
Load the authentication script ()
«HTTP»
opt Authentication process
[The user is not logged in yet]
:token (Ciphered User Token)
load the WSE main script ()
«HTTP»
:wse.js
load the Enhanced Address Book script ()
«HTTP»
:eab.js

Figure 3. EAB call flow (loading the hosting page)

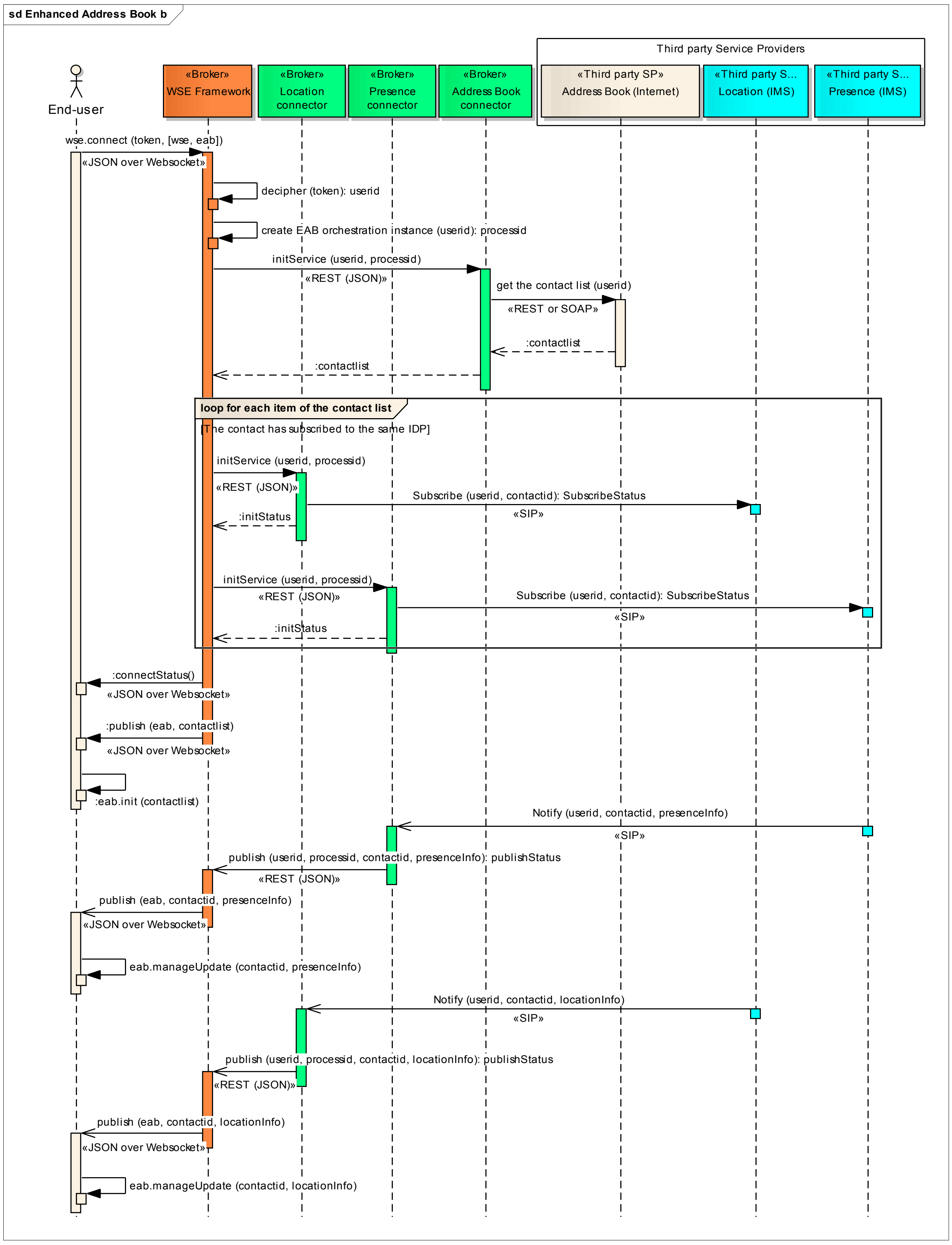
sd Enhanced Address Book b
Third party Service Providers
End-user
«Broker» WSE Framework
«Broker» Location connector
«Broker» Presence connector
«Broker» Address Book connector
«Third party SP» Address Book (Internet)
«Third party S... Location (IMS)
«Third party S... Presence (IMS)
wse.connect (token, [wse, eab])
«JSON over Websocket»
decipher (token): userid
create EAB orchestration instance (userid): processid
initService (userid, processid)
«REST (JSON)»
get the contact list (userid)
«REST or SOAP»
:contactlist
:contactlist
loop for each item of the contact list
[The contact has subscribed to the same IDP]
initService (userid, processid)
«REST (JSON)»
Subscribe (userid, contactid): SubscribeStatus
«SIP»
:initStatus
initService (userid, processid)
«REST (JSON)»
Subscribe (userid, contactid): SubscribeStatus
«SIP»
:initStatus
:connectStatus()
«JSON over Websocket»
:publish (eab, contactlist)
«JSON over Websocket»
:eab.init (contactlist)
Notify (userid, contactid, presenceInfo)
«SIP»
publish (userid, processid, contactid, presenceInfo): publishStatus
«REST (JSON)»
publish (eab, contactid, presenceInfo)
«JSON over Websocket»
eab.manageUpdate (contactid, presenceInfo)
Notify (userid, contactid, locationInfo)
«SIP»
publish (userid, processid, contactid, locationInfo): publishStatus
«REST (JSON)»
publish (eab, contactid, locationInfo)
«JSON over Websocket»
eab.manageUpdate (contactid, locationInfo)

Figure 4. EAB call flow (Presence and Location updates)

4. DISCUSSION AND OPEN ISSUES

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

4.1 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 [9].

Indeed, Websocket application servers do not need to wait for a Web client request in order to send new available data in a reply format. This is because of the full-duplex TCP connection which is established between client and server allowing each player to send data when it deems necessary.

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.

4.2 Simpler connections

Connectors can be developed and managed by the corresponding service provider, thus discharging the retailer. The protocol/data format conversion between service provider and retailer can be developed in any language. Hence, a cooperative model is established for providing services to end-users, and the WSE architecture is alleviated from the task of dealing with complex connectors tasks.

4.3 Standard technologies

In order to support asynchronous Web applications, we based our architecture on Websocket. This technology is a W3C and IETF standard, natively supported by most browsers, unlike other solutions which are proprietary and need specific plug-ins to run (e.g. Adobe Flash technology). All parties, end-users, retailers and services providers benefit from this advantage, as standard solutions make the implementation easier and native solutions do not need additional steps from the end-user.

4.4 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, as described in section 3.3. This entity is able to manage the composition of services regardless of their type.

4.5 Operator centric approach

The Websocket Enabler, if managed by the Telecom operator, allows for enhanced system security by using operator's authentication mechanisms to access services. In addition quality of service is ensured by operator's network and by Websocket technology in the Internet part of the connection.

4.6 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 [13]. 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*.

5. 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.

Because Websocket is an important player in our solution we will follow closely its standardization since this technology is still in a maturation phase where some security issues [10] are currently treated.

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 [7]. This added feature will allow further integration of multimedia services and their delivery through Web browsers.

6. ACKNOWLEDGMENTS

This work is supported by SERVERY[11] an European project of the Celtic initiative, whose goal is to enable a service Marketplace that bridges the Internet and Telco worlds by merging the flexibility and openness of the former with the trustworthiness and reliability of the latter.

7. REFERENCES

[1] Verdot, V., Boussard, M., Bouche, N., Shanmugalingam, S., and Fournigault, L., 'The Bridging of two worlds: a Web-IMS Communication Solution.' EuroIMSA, 2009, Cambridge, UK, July 13 – 15, 2009.

[2] Jain, M., Prokopi, M., ' The IMS 2.0 Service Architecture.' NGMAST, 2008, Cardiff, UK, September 16-19, 2008.

[3] Lozano, D., Galindo, L.A., García, L. 'WIMS 2.0: converging IMS and Web 2.0. Designing REST APIs for the exposure of session-based IMS capabilities'. International Conference and Exhibition on Next Generation Mobile Applications, Services and Technologies. Cardiff, Wales, UK, September 6-19, 2008.

[4] Thuan, D., Jonvik, T., Jorstad, I., 'Family Portal: combining IMS and the Web', ICIN, Berlin, 11-14 October 2010.

[5] Forestier Husson, F., Zakhama, N., 'Web Session Controller: an opportunity for IMS/Web convergence ', ICIN, 2008.

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

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

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

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

[10] Huang, L., Chen, E., Barth, A., Rescorla, E., Jackson, C., 'Transparent Proxies: Threat or Menace?', http://www.adambarth.com/experimental/Websocket.pdf, November 2010.

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

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

[13] 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.

[14] SOAP specifications, http://www.w3.org/TR/soap/

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

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