Providing Middleware Support for the Control And Co-ordination of Telecom Mashups
Nilanjan Banerjee, Koustuv Dasgupta, Sougata Mukherjea
IBM India Research Lab, New Delhi, India
Email: {nilanjba,kdasgupta,smukherj}@in.ibm.com

ABSTRACT
The emerging Web 2.0 marketplace presents an important opportunity for Telecom operators to sell their own capabilities and content as services. Operators have a wealth of content associated with their network as well as core network enablers, such as call control, presence and messaging, that could serve as potential new revenue streams in a Web2.0 world. Moreover, with the looming threat from Internet companies to build content and potential new revenue streams in a Web2.0 world. More over, with such communication paradigm are gaining prominence. Web – mode is one-to-one communication. Hence, technologies enabling such communication paradigm are gaining prominence. Web – based portals such as Orkut[10] and MySpace[11], as well as mobile-based presence-enabled services like Sprint Friend-Mapping[12] and BusinessFinder[3], facilitate such one-to-one communications between two interested parties. However, text or data has been the primary mode of communication in the majority these portals. Historically, voice has proved to be the most successful medium for one-to-one communications. The wide popularity of telephone systems followed by that of cellular phones provide ample evidence to this fact. Thus, it looks obvious that Web2.0 applications are missing an important medium of communication so far, and the applications would greatly benefit from any effort made towards enabling them with this mode of communication.

Bringing voice communication capabilities to web applications, however, is not new. Initially, SS7-based IN services were integrated with the Internet applications and several smart services (such as Click-to-Dial and IVR-based applications) were developed. However, the major problem with these applications was that they were based on SS7, which was designed as an in-band signaling protocol. This restricted the flexibility of the services that were developed based on SS7. The subsequent emergence of VoIP technology with Session Initiation Protocol (SIP) [7] as the out-of-band signaling protocol solved this problem to some extent by providing flexibility in the signaling plane. This in turn facilitated the development of advanced applications beyond the basic Click-to-Dial service, where it was possible to provide the call-control to the application developers. Although SIP provides the necessary flexibility, VoIP applications alone cannot leverage the full benefits that a traditional (cellular) telecom service provider offers, e.g. mobility, billing, and QoS, which makes them indispensible in building advanced services. However, it is a non-trivial issue to make these services available to application developers. The IP Multimedia Subsystem (IMS) was designed precisely to address this. Introduced by The Third Generation Partnership Project (3GPP) [1], IMS provides assistance and control for multimedia sessions, established between two communicating peers. More importantly, IMS promises to bring an entirely new set of media and collaboration capabilities to the next generation of “converged” applications, like voice instant messaging and video conferencing.

IMS defines a functional element called the Application Server (AS) in its service architecture to provide the value-added services. The AS’s in the network represent capabilities, which are system components that are used presumably with other components (e.g., content servers) to implement a service to the user. SIP is the IMS Service Control (ISC) interface used between the core network Call Session Control Function (CSCF) and the service capabilities implemented in the AS’s. The real problem and, in our minds, the gray area in the standardization lies in the interaction of these capabilities. The 3GPP standard only loosely defines the functional element called the Service Capability Interaction Manager (SCIM) that handles the interaction between several capabilities. The implementation details are left to the vendors. SCIM, however, falls short in bringing the IMS capabilities to the Web2.0 domain. For example, in a mashup environment, the SCIM needs to handle protocol level mediation between SIP and SOAP to merge the capabilities from two disparate domains of Web/SOA Services and IMS. In this paper, we address the need for an enhanced SCIM (e-SCIM) and present an architectural solution that implements the e-SCIM functionalities, which we deem as critical, for the enablement of telecom features in Web2.0 applications.
The rest of the paper is organized as follows. Section 2 discusses the requirements for introducing telecom services to the Web2.0 domain. Section 3 describes the IMS service architecture that acts as a key enabler for Web2.0-based telecom services. The proposed system architecture that enables the development of such services is described in Section 4. Section 5 presents a prototype application built to demonstrate the capabilities of the proposed architecture. Finally, Section 7 concludes the paper.

2. WEB 2.0 AND TELECOM MASHUPS

The World Wide Web is undergoing a fundamental paradigm shift, which is often referred to as Web 2.0. The Web 2.0 applications are delivered through Web browsers and the applications are responsive to users through rich application interfaces, often including pre-built application components or service widgets. Mashups are really the essence of Web 2.0 applications. A mashup is a website or application that combines content from more than one source into an integrated experience. So far the main focus of the mashup services has been the amalgamation of data from different applications. However, the architecture for data-based mashup services is not sufficient to bring telecom services into the mashup domain. This is because the Web 2.0 functional model is also supposed to be a Web Services model that blurs the line between software and services, and makes it complicated for complex services such as telecom services. IMS has definitely facilitated this process by defining a unified telecom service architecture which can be inducted into Web 2.0 mashup paradigm. However, there are some challenges that need to be solved before this happens.

![Figure 1: Trade-off between number of developers and service capabilities with service abstraction](Image)

A salient feature of Web 2.0 is to enable “long-tailed” applications by harnessing the collective intelligence of a wider base of developers empowered with lightweight API-based programming tools. The IMS way of developing new applications is a bit different from the Web Service based application development paradigm. IMS recommends SIP AS implementing the application logic that analyzes the SIP messages generated by a particular network service and take necessary actions. This protocol (SIP)-based service development in IMS enables rich services to be developed, which goes beyond the basic legacy services, in terms of capabilities. Examples of such services are those which are capable of being customized with rich, dynamic contextual information during runtime. As shown in Figure 1, the number of developers increases significantly when services are available as Web Services, but this also takes it toll on the service capabilities. The capabilities, however, increases when services are developed based on the protocols. Clearly, we do not want to sacrifice on either front. Hence, for telecom services to take off in a Web 2.0 world, we believe that middleware control and coordination is required to provide the flexibility of protocol-based IMS/SIP application development, as well as, the simplicity of Web Services to the mashup developers.

3. IMS SERVICE PLATFORM

The IMS reference architecture is that it segregates the service, control and transport layer. IMS, however, does not deal with the transport directly, but provides a mechanism to converge networks based on a broad range of different wireless or fixed access technologies. IMS uses SIP for the control of sessions. SIP is an Internet signaling protocol, which along with a suite of IETF designed multimedia transport protocols such as RTP, RTCP, provides complete support for multimedia applications. The main elements in this signaling plane are the SIP proxies or servers known as the CSCF servers. As shown in Figure 2, the service layer is completely decoupled from the control layer and the two interact via the ISC interface. This provides flexibility to new service logic without affecting the CSCF functions in the network layer. To interact with these proxies, the user devices must implement the functionality of a SIP user agent (SIP-UA). The CSCFs handle all the SIP session signaling, but they do not take part and neither are they on the path of the application data. The IMS proxies are hierarchically divided in two categories: (i) the proxies-CSCFs (P-CSCFs) which are the IMS contact points for the SIP-user agents (SIPUAs), (ii) the serving-CSCF (S-CSCF) which is the proxy server controlling the session.

Service invocation in the IMS is done by a trigger mechanism. The Serving-CSCF (S-CSCF) obtains the Initial Filter Criteria (IFC) from the HSS for a registered subscriber. When the subscriber sends a SIP command, INVITE for instance, for the multimedia session that one is requesting, the S-CSCF cascades through the IFC and sends the invocation to the necessary Application Servers based on priority. While the S-CSCF has been enabled to invoke multiple sessions in a sequence it is assumed that once the control is handed over to an Application Server, it will be the responsibility of the Application Server to coordinate with other services. Herein lies the problem. To implement a service that requires combinational services, such

---

1 The Long Tail Blog, http://www.thelongtail.com/
as those created by mashups, the control must shift between the two applications dynamically under the direction of the user. This also exposes the issue that the IPC can cascade through services, but does not allow interleaving on a dynamic nature.

The services are also known as “capabilities” of the network system components. As mentioned earlier, it is the responsibility of the AS to co-ordinate diverse capabilities in the network and handles the orchestration of the interaction between them. During the initial designing phase of the IMS, the latter functionality was assigned to a functional unit called the SCIM. However, the 3GPP standard did not specify anything beyond the following definition of SCIM [1]:

“The application server may contain service capability interaction manager (SCIM) functionality and other application servers. The SCIM functionality is an application which performs the role of interaction management. The internal structure of the application server is outside the standards.”

The implementation of the SCIM, however, has been left up to the vendors to suit their individual application requirements. Such a loose definition of SCIM and the lack of standard definition of the capabilities that should interact make the design of a generalized SCIM challenging. Assuming that the most basic instantiation of an SCIM is an AS that orchestrates multiple AS’s, we extend the functionality of SCIM to define what we refer to as an enhanced SCIM (e-SCIM). We envision that the e-SCIM would be empowered to take care of the additional requirements, such as user-driven service configuration with dynamic contextual information, imposed by new paradigms such as Web 2.0. The e-SCIM in this case behaves as a service blender that manages the dynamic interaction between service capabilities of IMS network expressed as SOAP-based services. It also handles the real-time mediation between SIP and SOAP messages. In this way, the e-SCIM facilitates the protocol-based service development. The e-SCIM implementing the above functionality forms the core of the middleware identified earlier, and described in further detail in the following section.

4. SYSTEM ARCHITECTURE

We propose here an architectural solution for the issues described above. Figure 3 depicts the proposed architecture, and is loosely based on the Web 2.0 architecture. Here we have shown the architecture for telecom vertical only and left the components required for content aggregation mashup applications outside the scope of this study.

The four main layers of the architecture are (i) Resource tier, (ii) Service tier, (iii) Client tier, and (iv) Mashup tier. IMS service architecture described earlier forms the core of our service and resource tier.

The resource tier consists of all the network elements that gives control to the various network level services such as call, location, presence, SMS, etc. IMS assumes that all these services are SIP based and makes them available through the IMS Service Control (ISC) interface.

The service tier is supposed to provide a service abstraction of the network level services. This tier consists of high level Parlay X Web Service interface for legacy network services to third parties. Parlay/OSA defined APIs [5] were one of the early efforts towards this service level abstraction, which enabled to tackle the issue of service development for heterogeneous network to a large extent. However, to integrate the services developed with Parlay/OSA APIs with other business process (e.g., OSS/BSS) the APIs were enhanced to adopt a SOA-based model and were exposed as Parlay X Web Services. Parlay X provides to all third party application developers Web Services-based access to legacy telecom applications including third party call, call handling, billing, multimedia streaming and control, messaging, etc. [6]. Apart from the Parlay X services, the service tier also consists of all other IMS services that are deployed in the AS’s and are exposed as Web Services. Besides, this layer also acts as the placeholder for future IMS-based services.

The most important component of the service tier is the Service Coordinator (SC). The SC actually implements the e-SCIM functionality discussed in the previous section. The primary functions of the SC are as follows:

(i) Registration of services
(ii) Invocation of services
(iii) Rule-based Coordination of services – both functional and business rules
(iv) Coordination of services through dynamic user interaction

The services coordinated by SC range from Web Services to SIP-based IMS services, which implement diverse capabilities in the network. A service, which wants to be a part of the coordinated service, contacts the SC and asks it to create a coordination context. The SC will create a new instance of the coordination context and hand the context over to the requesting participant service. A coordination context is actually a data-structure that contains an activity identifier. The requesting participant service will then ask the registration service of the SC to register its role in the coordinated activity. The registration service will register the role of the participating service in the activity. The participating service then might propagate the coordination context instance to other services that it would like to take part in the same activity.
The SC coordinates various activities that are invoked when certain events happen in the ISC layer. The SC is equipped with a service rule repository and a rules engine, which help it to take actions or invoke the appropriate coordinated activity against the events of interest captured from the ISC layer using the IPC. The rules actually maintain the mapping between the ISC events and the corresponding activity context to invoke the designated activity. They also implement the appropriate mode of invocation of the coordinated activities. For example, if a Web Service based billing is invoked based on a SIP-based call, the rules implement the appropriate SOAP-based invocation to the billing Services. This coordination activity of the SC realizes the SIP-SOAP mediation required for rich service interaction and user driven service customization, according to the Web 2.0 paradigm. The role of the SC would be further elaborated with illustrative description in the subsequent section.

The four main functional components of the client tier in our middleware and their respective functionalities are as follows:

- **Controller** – forms the core application master logic and processing
- **Virtual Machine** – provides the runtime environment launched and managed by controller
- **Rendering Component** – defines the behavior for GUI’s and media integration
- **Data/State Management Component** – keeps track of transformations, state synchronizations, transitions, state change event generation during lifecycle of objects such as widgets.

The client tier provides the tools necessary for developing the mashup applications using resources from service tier. The controller does the job of sequencing the various services constituting the mashup application. In mashups, user interaction is very important and the controller, with the help of the virtual machine and the rendering component, provides an interactive GUI to enable the user interaction. The data or state management component is responsible for all the book-keeping functionalities required to relate the state transitions or transformations happening at the mashup tier and the service tier.

Finally, the **mashup tier** consists of the mashup applications developed with support of the underlying tiers.

5. PROTOTYPE IMPLEMENTATION

5.1 Scenario

To demonstrate a Telecom mashup application we consider the following **Call-a-Cab** scenario. Linda is a tourist in New Delhi, India. After her tour of the historical India Gate, she wants to take a cab back to her hotel. However, Linda is unfamiliar with the city. Using her GPRS-enabled phone, she logs on to the Call-a-Cab portal and requests for a cab. The call-a-cab portal uses Linda’s mobile phone number to find all available cabs in the vicinity of Linda’s location and displays them on a map. Each of the cab is marked with a plurality of attributes like its Estimated Time of Arrival (ETA), Customer Satisfaction Ranking, etc. Linda then selects a cab according to her preference and places a call to the cab driver using a GUI widget (Call/SMS) embedded in the portal. Before the call is connected Linda is interacted to the premium rate at which she will be charged for the service. The cab driver (John) picks up the call - Linda notifies him of the exact location and waits for the cab. John then starts driving towards Linda’s location, and when it is in the vicinity, an SMS notification is send to Linda’s mobile number. Along with it, a feedback request is also send to Linda asking her to rate her experience with the cab driver. Once she reaches her destination, Linda could provide this feedback to the Call-A-Cab service.

5.2 Prototype Implementation Details

We have completed an initial prototype of the Call-A-Cab mashup service enabling the above scenario. The service is a mashup of three core components: (i) Google Maps\(^2\) (ii) BusinessFinder matchmaking service \[^3\], and (iii) SIP-based 3rd Party Call Control (3PCC) Service.

Figure 6(a) depicts a snapshot of the Call-A-Cab portal. Linda logs on using her mobile number and searches for a cab. BusinessFinder extracts the current location of Linda and matches her request to a list of available cabs that are located in the vicinity. The cabs are displayed on Google Maps. Further, clicking on a cab driver returns real-time contextual information (e.g., ETA). Linda can either call or SMS the driver, as shown in Figure 6(b). In case she decides to call, she is intimated of the premium rate that she will be charged at. Finally, when the call is terminated (“End Call”) by Linda, the portal sends a “Thank You” message back to Linda. This is illustrated in Figure 6(c).

![Figure 4: Execution Flow for the Call-a-Cab application](image)


driver, ETA etc. This information is forwarded to the Google Map service, which renders a map displaying the results and sends it back to the application via the controller. Once Linda clicks to call the driver, the controller invokes the 3PCC service provided by IBM Telecom Web Services Server™ [4] with the mobile numbers of Linda and the cab driver as parameters. Note that, whenever such an invocation is made to a backend service, the client tier (i.e. data/state management component in it) maintains book-keeping information to correlate a client service session with a backend service session. As we will see later, this plays an important role in the service co-ordination between the client mashup and the SIP service. The current implementation assumes that both the calling and called parties are equipped with a SIP terminal. Such terminals could as well be a SIP soft-phones embedded within the browser application of the parties. Once the SIP session starts, the Service Coordinator (SC) comes into play through the IFC configured in the CSCF servers. For example, in this case, the 3rd party call context (including the SIP URI for 3PCC) is used as the filtering criteria (i.e. IFC trigger) to filter appropriate SIP messages and trigger corresponding SC actions.

5.3 Role of the Service Coordinator (SC)

The SC plays the key role of bringing capabilities from both Web2.0 and telecom domains, based on the rules specified in the SC rules engine. The capabilities in the Web2.0 domain are managed by the controller in the client along with SC rules in the mashup tier. The telecom capabilities are invoked by the SC in the same way as described in the IMS standards. In our case, we have used Parlay-X services such as those provided by IBM TWSS.

5.3.1 SC Rules

The SC is pre-configured with rules specifying following actions:

1. Intimate the user of premium charging for the service
2. Start charging once the call gets connected
3. With the termination of the call, the negotiated charge is committed and a “Thank You” message is sent to the user.

Formally these rules are represented as Event-Condition-Action (ECA) type if-then-else rules, where each of the rules consists of a condition clause and an action clause that is executed when the condition is met. So the above rules can be represented as follows:

- **Rule 1**: If Call_Initiated then intimate user of premium charging
- **Rule 2**: If Call_Established then start real-time charging functions
- **Rule 3**: If Call_Terminates then commit the charges for the service
- **Rule 4**: If Call_Terminates then present “Thank You” message

The above rules, however, deals with normal case of call set up. The SC can additionally have rules on how to handle exceptions. The following two rules, for example, show possible actions that could be taken in case the call setup fails.

- **If Call_Fails** (not reachable, not found) then present a list of next available cab driver with necessary credentials
- **If Call_Fails_Provisionally** (driver busy, driver not picking up the call) then present an option for waiting for sometime or calling another driver

5.3.2 Conditions

The conditions in the SC rules are based on SIP events that take place in the ISC layer during the 3rd party call setup. Figure 5 shows the SIP call flow for the call setup between Linda and John’s devices. Basic Call flow for the 3rd party call is defined by RFC 3725 [2] Call flow III. The service uses two SIP servlets to control the third party call Back to Back User Agent (B2bUA) [8] implementation. The first servlet handles the initial part of the call (Messages 1, 2, and 3). Once the ACK is received from the initial steps, the SIP control is passed to the second servlet which handles the rest of the call flow.

![Figure 5: Call Flow for Third Party Call Setup](image)

The first rule condition corresponds to the INVITE message (Message 1 in Fig. 5) intercepted through the IFC interface in the CSCF server. The second condition gets satisfied when the call is established, i.e., ACK message (Messages 9 in Fig. 5) is sent to Linda’s SIP terminal. Finally the third condition is met when the call is terminated i.e., OK (Message 13 in Fig. 5) message is sent to John’s terminal in response to the BYE messages.

5.3.3 Actions

The SC leverages the client tier and other services in the service tier to implement the actions in the rules engine. The SC act to the events in the call and sends messages to the controller of the client tier - which then makes use of the virtual machine and the rendering component of the client tier to get the appropriate callback objects and send messages as pop-ups in the client browser. The controller takes help of the data/storage management component to find reference to the callback objects corresponding to the appropriate mashup service session.

The SC sends a pop-up to the client application with the charging details when Rule 1 is triggered. With triggering of Rule 2, the SC invokes the charging service from the service tier and starts real-time charging. Rule 3 makes the SC invoke the charging service again, this time with appropriate parameters to...
6. CONCLUDING REMARKS

Web 2.0 is proving to be a viral technology built on personalization, collaboration and participation. Innovative Web 2.0 companies are leveraging this environment to grow, unleashing the creativity of the developer community by allowing them to mashup internal functionality or content with one or more other sources, using easy, standards-based drop and play tools, to create new value-added blended applications. The Web 2.0’s service creation framework, with its ease of use, low cost and collaborative nature threatens the telcos’ walled-garden strategy, and potentially marginalizes the promise of IMS. Rather than competing against Web 2.0 companies, the telcos need to embrace their service creation and execution framework, allowing outside developers to easily and rapidly integrate their content with telco enablers, enablers that are most effectively done in the core and for which the telcos have a competitive advantage. These include functionalities such as location, presence, bandwidth policy, identity/authentication, QoS, billing (micro and other), messaging, profiling and more.

Nurturing such an ecosystem will allow service providers to develop and test new revenue generating services much faster and cheaper than they could ever do on their own. New revenue will also be realized through fees paid by third-party developers for network functionality access, billing and more. This paper presents our current efforts underway to capture and integrate the co-ordinated execution of next-generation Telecom services within the Web 2.0 framework, in order to efficiently execute and monetize new blended applications and next-generation services.

7. REFERENCES


