Appears in, *Proc. 2nd IEEE Int'l. Conference on Internet Multimedia Services Architecture and Applications* (IMSAA-08), Bangalore, Dec. 2008

Mobile Contextual Mashup Service for IMS

B.Falchuk, K.Sinkar, S.Loeb, A.Dutta
Telcordia Technologies, Inc.
Advanced Technology Solutions
Piscataway, USA
{bfalchuk|ksinkar|shoshi|adutta}@research.telcordia.com

***Abstract* — The Web 2.0 paradigm is all about the development of applications and content at the edge of the network by anyone. This "wisdom of the crowd" type of activity resulted in a large number of applications being introduced in short time frames utilizing the telecom network as a "dumb pipe". However, IMS may open up the business opportunities for the cellular operators where they can take advantage of the openness of APIs offered by Web 2.0 dot-coms and still offer the flexible call control features of IMS. Thus, convergence of Web 2.0, mashups and IMS call control features have suddenly become a viable business model that will benefit the end users, cellular operators and application service providers at the same time. Rapid deployment of IMS can certainly be enhanced by successful blending of mashups and built-in features of IMS such as security, authentication, and quality of service. However, there is a lack of existing use case scenarios and experimental validation of supporting various web 2.0 applications over IMS. In this paper, we provide such a framework that helps to blend these two and highlight the lessons learned from the prototyping effort. We believe this framework and analysis of the results obtained from this experiment will be illuminating to cellular operators who plan to deploy IMS infrastructure and offer mash-up type services as well. A new type of service called "contextual" mashup is also described.**

Keywords: Web 2.0, IMS, Mashup, Context Aware Mobile Services, Location Based Services, Proactive reminder system

I. INTRODUCTION & MOTIVATION

The Web 2.0 paradigm is built around creativity at the edge of the network by allowing third parties to introduce new services quickly and easily and treating the operator's networks as a "dumb pipes". Currently, several standards bodies (3GPP) [3] are defining the architecture for IMS (IP Multimedia Subsystem) that will enable current cellular operators to provide flexible services and richer call control features over an IP-based network. While many carriers have begun trial deployment of IMS, some are still concerned about the amount of complexity involved in IMS and lack of "killer" and open services that are currently offered by IMS.

A Mashup is an application that combines data from more than one source into a single service. There can be several types of "mash up" applications, such as consumer mashups, data mashups, enterprise mashups and business mashups [2]. As examples, combining cartographic data from Google Maps to real-estate data creates a new and distinct service that was not originally provided by either source. Similarly, a "telecom mashup" is a telecommunications service where service elements come from more than one source and are combined into an integrated experience. For example, one could get the base service from company A, a ring-back tone from company B, a voicemail service from company C, etc. A mobile "Contextual Mashup" extends the concept of "telecom mashup" by combining elements of a user's digital life together [7]. It is becoming more straightforward for end-users to create mashup sites regardless of their technical skill level by using a site's Application Programming Interface (API). It is now widely thought that the deployment of IMS can benefit greatly from the blending of its rich call control features with the existing Web 2.0 or "mash up" applications that are based on open API. This paper expands and supports this idea: our Web2.0/IMS testbed together with the innovative notification services we envisage, are both novel *and* of current interest.

A. Business Models - An Opportunity for Operators

Many cellular operators consider these new paradigms as opportunities to build business cases in which they can enable IMS infrastructure and provide the existing "mash up" services on the top of it. By offering IMS capabilities to the Web 2.0 world, these can be included in the Internet mashups. Similarly, exploiting the Web 2.0 services and technologies can enrich the operator's services. Although IMS infrastructure provides inbuilt support for security, quality of service and call control, it needs certain modifications in some of the components such as user agent, application server in order to support this type of application. Thus, it is useful to explore the feasibility of supporting a mobile contextual mash up service over an IMS platform and explore the methodology involved.

In order to create innovative, appealing and user-centric services and applications it is useful to have a convergence between Web 2.0 and the new generation services and call control features offered by IMS. While the former provides relevant characteristics such as interactivity, ubiquity, social-orientation, user participation and content generation, the latter provides a platform to complete the service features with multimedia telephony, media sharing, push-to-talk, presence and context all applicable to mobile, fixed or convergent telecom networks. In addition, IMS infrastructure provides additional features, such as user authentication, security, call control, quality of service are very important and pave the way for generating revenues by the carriers. Thus, a business model based on supporting mash-up application over an IMS

platform can benefit the end-users, carriers and third party application developers.

B. Related Work

Carriers are currently trying to build business models around new Web2.0 and collaborative technologies. Banerjee et al [9] discuss a possible architecture involving telecom mash-ups. However, it does not use IMS as the infrastructure to take advantage of the associated benefits. Shin et al. [8] describe the end-user driven service creation method for converged service over telecom and Internet domain. It focuses on service layer and takes advantage of the open network architecture but does not utilize IMS's rich call control feature. Our work takes advantage of the flexibility and openness of Web 2.0 framework and blends it with the native call control features of IMS. Elsewhere, off-the-shelf event engines (e.g., Coral8, IBM, etc.) provide native handling for streamed events and message routing. These systems are not geared directly towards supporting time and location while we propose several algorithms that help us achieve scalable mashups for operators as described in section III and [7]. Single-purpose Web 2.0 reminder systems have emerged online [4],[5], and while they are a good start they may not scale well for millions of customers, show little inferencing capabilities, and have little ability to use contexts from diverse or subtle aspects of a user's life.

The remainder of this paper describes our live IMS testbed as well as the related features and challenges. We then describe the functional components that implement a scalable event mashup engine called Proactive Services Platform (PSP) and describe a typical mashup service. We conclude with lessons learned. Hereafter, we focus mainly on the testbed components and the general value of notification mashups rather than mashup service logic.

II. IMS TESTBED

In this section, we describe the experimental IMS testbed and the functional components that support many of the basic features such as registration, multi-media call setup, security association, quality of service, charging and mobility. The testbed is unique because it enables SIP-to-non-SIP interaction and advanced features such as voice-call /IPTV interworking [1]. Besides operators, there are not many fully advanced testbeds comparable to ours (e.g., Georgia Tech, GMD Fokus.)

Figure 1 shows parts of the experimental IPv6 testbed with different functional components that we have prototyped. These include many of the standard IMS components such as P-CSCF(Proxy Call Session Control Function), I-CSCF (Interrogating Call Session Control Function), and S-CSCF (Serving Call Session Control Function) for signaling, Home Subscriber Service for registration function, Home Agent for the bearer manager functions, PCRF (Policy Control and Rules Function) for policy control and charging functions, SIP [10] Application Server for enabling SIP related application, PDSN (Packet Data Serving Function) and PDIF (Packet Data Interface Function) for access networks, IPTV server for non-SIP-based services. Event Generator and HTTP proxy are the additional functional components that interact with IMS components to realize (and test) a Web 2.0 Application. Section IV describes how some of these components are used to provide a Web 2.0 application over IMS platform. The rest of this section outlines the key aspects of the testbed.

*Radio Access Networks***:** Currently, we have configured two kinds of radio access networks in the IMS testbed, such as Wi-Fi (802.11) and CDMA2000. In the absence of real CDMA networks, we have emulated CDMA2000 using PPPoE and RAN emulator. Thus, the current testbed provides the ability to support the handoff between Wi-Fi and PPP networks.

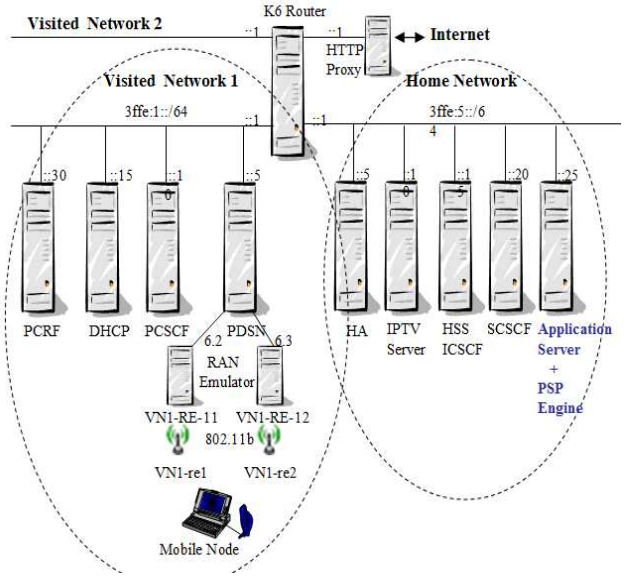

Figure 1. Experimental IMS Testbed

Address Configuration: Since the end hosts are IPv6-enabled, we have implemented both PPP and stateless modes of auto-configuring the IP addresses. In stateless mode, the mobile configures its IP address based on the prefix of the router advertisement. In PPPoE mode, the prefix is provided to the mobile by the PDSN.

Server Discovery: Server discovery is a process of discovering the addresses of the outbound SIP servers (P-CSCF) in each subnet. After configuring an IP address, the mobile uses DHCP INFORM message to obtain the IP address of the server. As the mobile changes its network, it rediscovers a new P-CSCF in each subnet. SIP registration (re-registration) and SIP INVITE (re-INVITE) use the newly discovered proxy to send these messages.

Registration: Registration is a process by which the mobile communicates with the network to establish its new association with the network. The mobile is a SIP client associated with a unique URI (Universal Resource Identifier). It registers with the HSS (Home Subscriber System), via the current P-CSCF in the subnet. Registration message actually travels via I-CSCF and S-CSCF. S-CSCF and HSS communicate with each other using Diameter protocol.

Multimedia Call setup: Since SIP is used as the signaling protocol, the mobile users can communicate with

each other using SIP URI. Both the mobile users update their updated address-of-records and care-of-proxy servers with the S-CSCF in the home domain. If MIP (Mobile IP) is used, home address is used as the address-of-record and the mobile just uses the new outbound proxy server. Thus, any new call gets routed via S-CSCF, HA, and P-CSCF before reaching the callee. But, if application layer mobility is used, then the mobile uses new care-of-address as the new address-of-record during SIP registration. In the absence of HA, the new call gets routed to P-CSCF directly without being routed to home.

Security Association: AKA (Authentication and Key Exchange) is a way of making sure that the mobile is authenticated and there is a security association between the mobile and the P-CSCF. AKA procedure is performed during mobile's SIP registration with S-CSCF. During the AKA procedure, the security association is established between MN (Mobile Node) and P-CSCF. Thus, a user with Web 2.0 application can benefit from added security association using the AKA procedure.

Quality of Service Support: Quality of service is provided by way of interaction between PCRF, P-CSCF and PDSN or PDIF. Once the security association is established, PCRF is informed and the gating on the PDSN or PDIF is turned on by using Diameter protocol. Bandwidth parameter associated with any specific media (e.g., audio or video) is contained in the SDP (Session Description Protocol) part of SIP message that is parsed by the P-CSCF.

Application Server: We have integrated Application Server IBM Websphere 6.1 in the home network. WebSphere Application Server delivers rich SIP functionality throughout its infrastructure. Using the application server, one can write applications utilizing the SIP Servlet 1.0 specification. WebSphere Application Server also provides tooling for the development environment and high performing edge components to handle distributed application environments. The SIP components in WebSphere Application Server have a tight integration with the existing HTTP servlet and portlet work, with which one can write a highly converged HTTP and SIP application with seamless failover provided by the Proxy Server. In our case, It interacts with S-CSCF via ISC interface and communicates with HTTP proxy using HTTP interface to carry the location-based information.

HTTP Proxy: In our testbed HTTP proxy is a dual stacked system with support for both IPv6 and IPv4. It interacts with the Internet using IPv4 backbone and communicates with the Application Server on the IPv6 side using http interface.

Charging Support: Cellular providers can use IMS infrastructure to charge for any specific type of "mashup" application. Details about the type of application and duration of call can be easily retrieved from the P-CSCF and PCRF can store this context.

Event Generator : A Java-based testbed component (Figure 2) that greatly simplifies testing, experimentation, and demonstrations, such as emulating a mobile user. It allows the single or scripted (batch) emission of custom events into the Proactive Services Platform (PSP) core. Types of events that can be emitted include: GPS locations (e.g., as if from a location provider or mobile user), to-do and calendar details (e.g., as if from a calendar provider such as Yahoo! or Google), Telematics info (e.g., "low fuel" as if from an on-board unit), traffic, weather and so on. Each event is emitted conforming to the appropriate XML Schema.

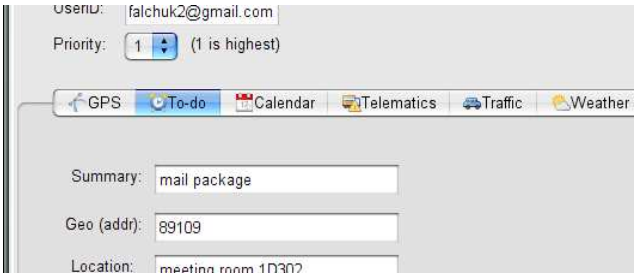

Figure 2. Event Generator tool emits events into the core; e.g., emulation of contextual changes such as location, new to-do entries, appointments, etc.

Mobility Support: Current version of IMS testbed can provide both network layer and application layer mobility support. As part of network layer mobility support, we have implemented Mobile IP v6, and as part of application layer mobility we have implemented SIP-based mobility [10]. The mobile can either be connected to a PDIF over an 802.11 network or PDSN over PPPoE (PPPover Ethernet). This mobility support can provide seamless connectivity to any type of "mash up" application when the user moves from one access network to another.

In summary, the IMS testbed is a robust operational environment in our labs that allows us to create and test new and innovative IP and telephony-based services in a highly mobile environment. The next section describes an all-IP service referred to as a "contextual mashup", and the related server-side concerns and architecture.

III. PROACTIVE SERVICES PLATORM SUPPORTING CONTEXTUAL MASHUP

Given both the changes and advances brought on by Web 2.0 and mobility, there is a new and undeniable trend towards intelligent notification services. Such a service can be thought of as one that provides the "right information at the right time" (RIRT) to a mobile user. RIRT has long been the pretext of military and emergency systems where informational advantages can directly relate to strategic and tactical advantages. Today's highly capable mobile devices, higher speed edge networks, and open API's are the right brew to deliver RIRT and improved mobile experience to the mass market end-user to help with today's busy lifestyle. In early 2G networks, mobile services had little scope beyond communications (e.g., SMS, voice sessions). 3G networks saw the deployment of many kinds of new services: standalone (binaries), streaming services, and mobile versions of many of the most popular Web sites. The iPhone enabled a revolutionary increase in user experience and will likely foster new services such as: immersive mobile applications, virtual location-based assistants, and proactive services that take control of life-details that users find boring or repetitive. Steps in this direction include *RememberTheMilk* [5] and

OhDontForget [4], and a variety of other services for mobile that, to some extent, manage personal reminders and calendars. They enable users to enter tasks with explicit task-locations; later, when on-the-go and nearby a task-location, the service notifies the user (e.g., "You are near the Mall where your task, 'Buy a shirt' can be completed".) The authors' own research comprises a much more proactive, intelligent approach [6][7]. The Proactive Services Platform and context mashups it enables are different because it:

- Extensible, separate logic (for new service creation)
- Doesn't require explicitly pre-entered reminders – will make some inference about when and where
- Resolves semi-unstructured task and appointment details (e.g., "mail a letter today") into a notification that makes semantic sense (e.g., a post office).

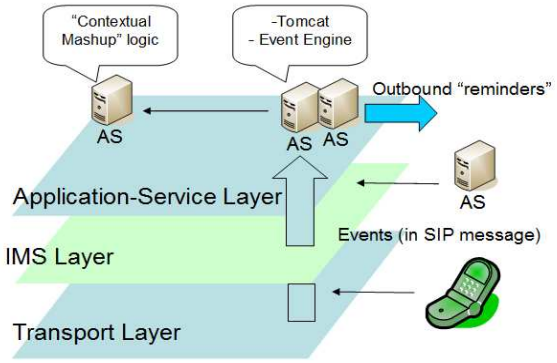

Figure 3. High-level positioning of the Web2.0 over IMS application.

Figure 3 illustrates the positioning of a contextual mashup service within the IMS "stack". Events are sent inbound via SIP messages from mobile devices and Application Servers (AS); reminders and other service logic are ultimately computed in the application-service layer. The next section describes the components in more detail.

A. *Proactive Services Platform*

The Proactive Services platform (PSP) is a software platform intended for the server side. PSP supports scalable event-based mobile services, such as intelligent, location-based, proactive reminder systems; however any system exploiting events can be implemented upon it. PSP is architected to take care of the following two main concerns:

- Scalable event handling and routing, storage such that events from many thousands of subscribers as well as scores of service partners can be managed
- Services, service logic, communications with the subscriber-base (in practice, thousands of users)

PSP is intended to handle incoming events (over various transports). The events comprise bits of user or system 'context' and, when put together, allow PSP to offer services that exploit the global view. Events have informational dimensions as well as location; e.g., a road traffic event has a location and has a traffic semantic (e.g., class: *traffic*, condition: *heavy*, location: *Rt.17, exit 5*). They have a scope that can apply to a single end-user (e.g., a user-location change), to a group, or globally (e.g., weather conditions for the East Coast). By way of scalable handling, storing, and understanding of event streams, service logic can be triggered at the appropriate time and a "contextual mashup" can be built that effectively provides a proactive reminder system to end users by "mashing together" different streams of user context.

The PSP core components reside in the IMS Application-Service Layer. PSP may interwork with other components providing session and communication capabilities on the IMS network but the main concerns of PSP are event handling and service logic triggering. In order to handle events forwarded in a scalable manner, the PSP examines incoming event metadata via an efficient pull-parser. If the incoming event has no location or time relevance to any existing customer then it is not (immediately) emitted into the in-memory event context. The *Esper Complex Event Processing Engine* (esper.codehaus.org) – or any comparable engine – can be used as the basis for the in memory structures. Our PSP system maintains an in-memory, rapid-lookup, data structure of the global aggregated user context, indexed by time. For example, as an event arrives, its time interval is scanned and compared to the same interval of the customers global context; whether or not that interval is "clear" of existing events (indicated by a Boolean flag) helps the system very rapidly decide on how to route and manage that incoming event. Similarly, a comparison with scalable, geospatial data-structures (e.g., quad-trees) that capture the presence of events in particular locations are used to allow the system to rapidly determine the impact of incoming events on existing ones with respect to location.

1) *"Contextual Mashup" Use Case*

Inasmuch as PSP platform enables contextual mashup services to be built for customers, the following is a system describing end-to-end use case: 1) user subscribes to service, 2) users context-providing systems are "redirected" to emit event into PSP (via HTTP), or PSP is configured to "pull" it, 3) as user context changes (e.g., a location provider emits user GPS cords into PSP on intervals) PSP builds a contextual database, 4) as user enters tasks (e.g., "mail that package") and calendar appointment they are stored in PSP, 5) when the current user context indicates a proactive notification should be sent to the user (i.e., application-specific service logic) the PSP arranges an outbound message (or a user Web portal is updated and the user "pulls" it into the mobile device Web Browser).

IV. IMS "CONTEXTUAL MASHUP" INTEGRATION FRAMEWORK

In this section we describe the integration framework with IMS. Events generated by the Event Emulator are wrapped into MESSAGE method and sent to the application server via. P-CSCF and S-CSCF. Security association of client with P-CSCF ensures that private information related to an event such as location is secured. Application server may also use location server to get an approximate location of the user if

client's location is not included in the event message. Application server extracts payload of an event message and forwards it to the PSP engine.

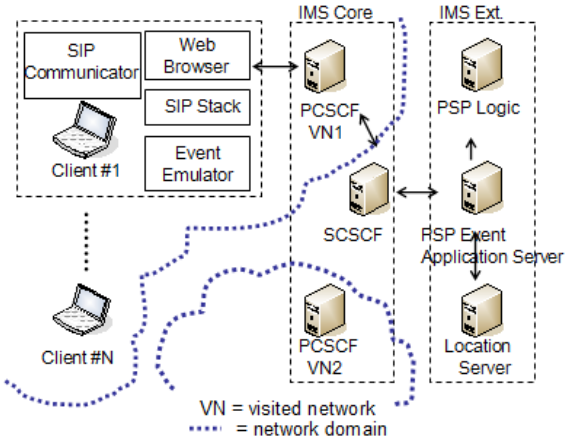

Figure 4. The integration of the PSP core into IMS realm. Events from the handset (Client #1) are wrapped in SIP messages and sent to P-CSCF/S-CSCF then forwarded to the application server where they are treated as context

We have used Java based open source soft-phone called NIST SIP-communicator as shown in Figure 4. It is modified to support IMS client functionalities such as AKA security and session mobility. For the "*Reminders*" application, the SIP-communicator is a SIP client used by both the mobile handset (e.g., laptop) and the external systems (e.g., calendar systems) that emits SIP messages towards the server. We can broadly divide functionality of IMS client into: 1) SIP stack is a light weight NIST stack responsible for registration, call handling and messaging. 2) Web browser for the client to connect to the "contextual mashup" portal, 3) Event Emulator generates events with attributes such as type, location, etc. and formats the data into an XML document describing the context, 4) SIP communicator implementation. The client uses the messaging module of SIP stack to send generated xml event schema as "txt/xml" payload of MESSAGE [3] method to the service running on Application Server. A Java Server Page is rendered by backend Tomcat http server when the client connects to the portal. The client uses a hook in the SIP stack to filter out "reminder service" messages received from the Application server and backend logic.

A. Compression of SIP Messages

A lesson learned during experimentation was that certain types of incoming events, e.g., calendar events complying with iCal schema, were longer than the others. On some of these rich events their XML representation grew longer than the allowed length (1300 byte) that a single SIP MESSAGE method can carry as defined by RFC 3428. To accommodate this shortcoming, we applied the gzip algorithm built into the Java2 SDK and ensured that event context was compressed before embedding into the SIP message and ungzipped at the receiving side. This solved our platform problems but in general message/event length could cause additional overhead and coordination requirements on the server side if they must be broken into a series of SIP messages rather than just one.

B. Results and Demo Scenario

We have implemented the proactive notification on the IMS testbed. We have used both Linux laptops and Windows Ultra-Mobile PC's to play the role of mobile devices (customers). We have also used Google Maps and Yahoo! Local Search to find local businesses when necessary. We also use Yahoo! reverse geocoding and integrate with Microsoft Outlook and Google GCal.

Figure 5. UMPC emulating mobile device for the customer in the Lab. The PSP platform sends intelligent reminders at the right moment.

In figure 5, the emulated customer uses the "lightweight" prototype device that simply uses a Web browser. By using a data connection in the Lab, a mobile UMPC user checks her portal (Figure 5), which is constantly updated by the contextual mashup service (via Java Server Pages). On that page the user sees one's given reminders (green panel on right), a map showing one's own reminders and past locations (left), and the explanations of why suggestions have been made (in the callout bubble). On our testbed, for example, the contextual task "refill prescription today" resolves to pharmacies around the user's current location ("CVS Pharmacy" is being presented to the user in the "bubble"). User location is constantly streamed into PSP within SIP messages, as are calendar and to-do context.

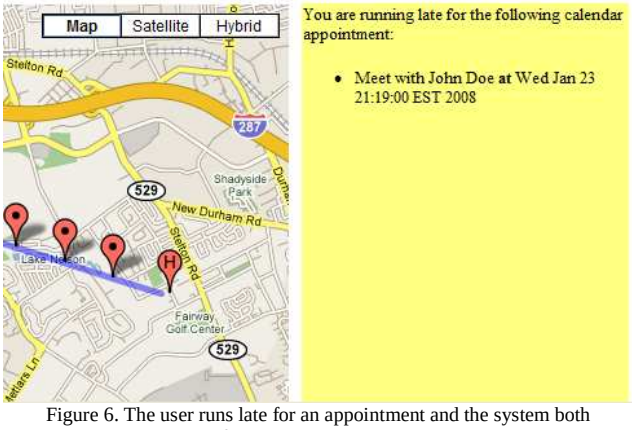

Figure 6. The user runs late for an appointment and the system both recognizes the fact *and* adapts subsequent interactions

In figure 6, the PSP system proactively sends a reminder to the user – not a commerce-related one (e.g., a pharmacy stop) but a time-based calendar notification. In this scenario the PSP has used two aspects of the user's context: future calendar appointments, and current and past locations. From the latter it computes speed and estimated arrival time to the former. If the user is deemed to be "running late" (inferred from current speed and direction and appointment location) the user is informed, while other less important reminders are suppressed, as in Figure 6.

C. Device and End User Issues

In the tested and initial implementation of the "contextual mashup" service we have assumed the end user has access to a Web browser as it is from the PSP Web portal that the user sees her relevant notifications. In this sense the service data must be "pulled" by the user (by starting the Web browser). The different variations of the client may therefore include: (1) Web-browser based service; customer needs only a data connection and browser and a URL to load. No events are generated automatically from the end device; the user manually inputs as necessary (e.g., appointments). (2) On-device binary application; a device resident application reads device data and automatically emits events into the core server (e.g., reads GPS coordinates and emits them). The server may either post info to the Web portal, or communicate directly with the binary application through a socket (either way a data connection is needed), or (3) SMS only; the customer and server communicate via SMS messages.

In general, the Web browser-based service is most portable while the on-device application is most powerful. Combinations of the above may also be viable.

D. Future Mashup Applications

While a "context mashup" service that delivers intelligent notifications is timely and novel, it is not the only type of service that IMS operators should be interested in. As hinted in the Introduction section and throughout, there are many new paradigms and services possible that combine elements of social networking, location, and communications. We are looking into exploiting the session control features of the IMS SIP servers to do things like dynamically create 3rd party voice sessions, interleave media sessions, and react to network conditions. These topics comprise future work and will be reported in the future.

V. CONCLUSION

In our Advanced Technology Solutions Lab, we have built, tested, and operated a full-blown IP Multimedia Subsystem (IMS) testbed on which operators and researchers can test innovative new services. With the growth of heterogeneous 4G networks, Web2.0 and the great promise of all-IP services supported by IMS, a perfect storm of technologies now enable network operators to get into the game. Most agree that the hottest and most promising mobile Web2.0 applications will likely be: mobile search, advertising, social networking, and reminder systems. To that end, we have integrated our innovative, scalable, server-based intelligent reminder system onto the IMS testbed as an exercise of integration and deployment.

Supported by event processing algorithms innovative service logic, the intelligent reminder system becomes an all-IP service in which user context and information is propagated inbound via SIP messages and value-adding computations on global context supply end-users with intelligent "Reminders", such as: "don't forget to mail the package to dad today". These reminders are based upon user context previously propagated inbound (e.g., in the above example a user-to-do item was entered into a Google calendar from where it was pulled via RSS into the core server. Alternatively, SIP messages emanating directly from the user device conveying location (GPS), telephony (session) and calendar info are also explored.

Our results dovetail with the *Wims 2.0* goals [2] inasmuch as they blend Web2.0 and IMS in mutually beneficial ways. Our integration makes Web 2.0 more mobile within 4G all-IP network and proposes SIP as Web2.0 transport. It also adds business value to IMS by showing the possibilities of Web 2.0 style services for mobile users.

Our experiments and deployments of this contextual mashup were a success and future work includes further exploitation of the SIP protocol, and investigation of what other types of session-based services would be valuable to 4G operators.

VI. ACKNOWLEDGEMENTS

Authors acknowledge collaborators Tsunehiko Chiba, Satoshi Komorita and Hidetoshi Yokota of KDDI Labs for useful discussion and feedback in this area.

REFERENCES

[1] A. Dutta et al., "A-IMS Architecture Analysis and Experimental IPv6 Testbed," *Proc. IEEE IMSAA 2007*, Bangalore, 2007

[2] Telefonica WIMS2.0 – Convergence of Web2.0 and Telco World, http://www.wims20.org/

[3] IETF SIMPLE Working Group, http://www.ietf.org/html.charters/simple-charter.html

[4] Oh, DontForget!, available at http://ohdontforget.com

[5] Remember the Milk, available at http://www.rememberthemilk.com

[6] B.Falchuk, S.Loeb, "Towards Guardian Angels and Improved Mobile User Experience", *Proc. IEEE GLOBECOM 2008 Communications Software and Services Symposium*, New Orleans, 2008

[7] B.Falchuk, S.Loeb, T.Panagos, "A Deep-Context Personal Navigation System", *Proc. ITS America 15th World Congress on Intelligent Transportation Systems*, New York, 2008

[8] T. Koskela et al, "Towards Context-Aware Mobile Web 2.0 Service Architecture", *Proc. UBICOMM'07*., Papaete, 2007

[9] N.Banerjee, K. Dasgupta and S. Mukherjea, "Providing Middleware Support for the Control and Co-ordination of Telecom Mashups", *Proc. USENIX Middleware 2007*, Newport, 2007

[10] J. Rosenberg et al., "SIP: session initiation protocol", RFC 3261, Internet Engineering Task Force, June 2002., http://www.ietf.org/rfc/rfc3261.txt