ABSTRACT
This paper describes the concept of service blending—the ability of two or more services to interact under a unified control scheme. We introduce a novel infrastructure component, Service Broker, designed to support efficient blending through dynamically loadable program modules called steplets. We demonstrate their use through implementation of two IPTV-based service examples.

Categories and Subject Descriptors
H.4.0 [Information Systems Applications]: General, Communications Applications.

General Terms
Design.

Keywords
Service Blending, IPTV Middleware, Service Broker, Service Infrastructure.

1. INTRODUCTION
The universe of electronic communications has been made up of separate worlds, in which specific services are associated with distinct networks. For example, the Public Switched Telephony Network (PSTN) has long carried phone calls, while TV networks deliver television programs and the Internet enables exciting data services.

Today, individuals desire to break through the separation of these communication worlds. They want their communications experience to be richer and more integrated. Consumers expect the ability to communicate anytime, anywhere using whichever device and method they choose. Instead of having to get up from the comfortable TV chair just to see who is calling on the kitchen phone, users would prefer to control the phone – or any other communications device—from the comfort of the TV chair, just by pushing a button on the TV remote control. They want to control their TV, personal voice services, Internet, e-mail, and Instant Messaging (IM) on whichever device they use, wherever they are, whether at home, at work or on the go. They want personalized, intelligent and seamless interactive features that can enhance and simplify their lifestyles.

The trend toward converged services and integrated applications is not new. The World Wide Web (WWW, Web), for example, makes switching between applications such as file transfer, streaming video, or e-mail seamless and effortless. The ease with which users can navigate through different applications and content contributed significantly to the tremendous success of the Web. Recent advances in networking and the trend toward packet-based infrastructures enable even tighter integration—the convergence and growing together of traditionally separate communication worlds. Voice-over IP (VoIP) carries phone services over the Internet, and IP-based TV (IPTV) promises to deliver television shows over packet-based networks. This broader convergence is transforming the communications industry, driving growth and creating new exciting services and revenue opportunities.

Service Bundling vs. Service Blending
Service providers are beginning to introduce IPTV offerings as part of a competitive service bundle designed to give consumers a flavor of service convergence. Service bundling offers unified ordering and billing, and limited interaction for otherwise separate services. It is only a first, limited form of convergence. Although it may help retain some customers in the short term, bundling continues to rely on separate networks for each type of service and does not provide the unified control consumers demand. Service blending, in contrast, enables different applications to control one another. It enables the Web or the phone to control the TV and vice versa.

The following example illustrates the difference between service bundling and service blending: If a user is watching TV and receives a call on her legacy phone, the name of the caller (i.e. the Caller ID) can be overlaid on the television screen using relatively simple technology. While this application provides a limited form of convergence, it is still just a bundled service—the Caller ID service (i.e., a voice service) is overlaid on a TV service.

The transition to a blended service occurs when individual services interwork and control each other. One such blended service might automatically record and/or pause the currently viewed TV program when the user picks up the phone to accept...
an incoming call. Later, when the user hangs up the call, an option to replay the TV program from where the user left off can be provided in form of an alert message on the TV screen. The seamless interplay of voice services (Caller ID) with TV viewing, video storage, and messaging (the replay alert) represents a truly converged service.

Blending IPTV services with telephony and Web applications is an opportunity to move beyond simple service bundling to create profitable, personalized, blended lifestyle services. In fact, tight integration is the key to true convergence. True convergence is a matter of uniting diverse applications, systems and technologies to build a flexible, scalable service infrastructure—one that can control voice, data, and video while providing security, reliability, and superior quality.

Unified Control

True service blending presents demanding new challenges: It requires infrastructure components that provide unified control among a variety of different applications—Web, IPTV, telephony. These components need to handle and map between the different state and data models underlying each of the services they blend together.

For example, an infrastructure implementing the blended service described above needs to handle both the call state model of the voice service and the state model of the IPTV application. It then has to map events in one model (e.g. incoming call accepted) to the appropriate command in the other (e.g. start recording). In addition, it has to map between the communications protocols that are used by the underlying applications—e.g. the Hypertext Transfer Protocol (HTTP), the Session Initiation Protocol (SIP), and the Real Time Streaming Protocol (RTSP). In our example above, an incoming SIP message triggers an RSTP message that initiates the recording of the TV program. Furthermore, the prevalent media encodings and transmission protocols of telephony are distinct from those on the Web. Therefore, the blending infrastructure must contain both signaling and media gateways for these networks.

The infrastructure must also maintain mappings among the various user and service identities that are used by applications in different networks. Finally, the infrastructure must contain elements to coordinate the operations of the applications involved in blended services. It must ensure that the interaction of multiple services does not result in undesired behavior. This is of particular importance since different services are likely not to be aware of each other.

Basically, the infrastructure has to provide an interworking layer that is able to support concurrent and coordinated access to multiple subtending applications. We design and present the Service Broker as the element implementing such layer.

2. INFRASTRUCTURE

It is important that the infrastructure used to build blended service is based on open standards and open interfaces. This significantly simplifies the integration of third-party services and allows tapping into the vast pool of existing applications.

The World Wide Web with its open architecture and standardized protocols proved to be tremendously successful in enabling the blending of non/near-realtime, content-driven applications. The IP Multimedia Subsystem (IMS) is an emerging architectural standard [1] that provides a promising foundation for adding support for blending of realtime, interactive services, including telephony and interactive multimedia. While IMS defines a comprehensive framework for voice and multimedia applications, it falls short in detailing the interworking with existing communications infrastructures such as the Web or IPTV. We later on address this issue by introducing our Service Broker, which serves as a bridging element between IMS, the Web, and IPTV.

2.1 IP Multimedia Subsystem (IMS)

IMS specifications describe a suite of functional elements communicating through standard protocols within an IP core network. This core network is typically surrounded by various access networks—including telephony networks and IP networks.

An IMS contains call session control functions (CSCF) that coordinate calls and sessions among other IMS’s, access networks, and applications. CSCF session controls are based upon the Session Initiation Protocol (SIP). IMS signaling gateways convert call and session events in other networks to/from SIP. Media gateways handle the formatting and transmission of data between an IMS and other networks. An IMS also contains subscriber data servers that store user and service descriptions; these data permit mapping of services and subscribers among multiple networks and applications. While the CSCF provides basic coordination among applications, a separate coordination server, called the Service Capability Interaction Manager (SCIM) may be present to provide more complex interaction management.
communicates with one or more application servers to coordinate their behavior according to the requirements generated by blended service specifications.

2.2 Service Broker and Steplet Concept

The Service Broker communicates with the CSCF through SIP messages and may treat some of these messages as call event triggers. As indicated in the introduction, we have extended the concept of Service Broker beyond the IMS SCIM, allowing our Service Broker to communicate with application servers through various protocols, including HTTP and SOAP. It can use these communications to respond to application-specific information, e.g., calendar updates. Therefore, the Service Broker acts as a programmable signaling gateway between different kind of networks and applications [2].

The Service Broker’s message handling logic is controlled by dynamically loaded program fragments named steplets, which are written in the Java programming language. Steplets determine policy issues, such as which application servers messages should be forwarded to, and how the messages that those servers generate should be dealt with. The Service Broker provides the mechanisms for functionality such as managing messages, invoking steplets, following the various protocols (e.g. SIP, HTTP, SOAP/XML), and maintaining transaction and dialog state.

Steplets are to the Service Broker what servlets are to a Web server servlet engine. A key difference, though, is that each HTTP request is typically handled by a single Web servlet. When the HTTP request arrives, the Web server invokes the appropriate servlet. That servlet may use other classes, of course, but ultimately that original servlet is responsible for generating the response to the client.

In the Service Broker, however, an incoming message may be handled by a succession of steplets—hence, the name steplet. In some cases, the sequence of steplets might be determined only upon arrival of the first message. For example, when a SIP INVITE message arrives, the Service Broker might look up the “To” address in a database, and the database would specify a fixed list of steplets to handle INVITE messages to the specified endpoint. In other cases, the sequence of steplets might be determined dynamically. For example, the Service Broker could look up the “To” address and get the first steplet. That steplet might forward the message to a specific application server, and, based on the server’s reply determine the next steplet to handle the message.

Steplets are similar to SIP Servlets [3]. However, their details are fundamentally different. While SIP Servlets are based on generic servlets as defined in the Java Servlet Specification [4], steplets use a different underlying model. We have chosen a different model because the Java Servlet Specification does not provide some features we see as facilitating Service Brokering and because Java Servlets obligate us to implement potentially expensive, unneeded features. Further, although the Java Servlet Specification was intended to be protocol-neutral, in reality the generic servlet classes are biased towards HTTP-like protocols. For example, the generic servlet classes assume that requests and responses are stream oriented, that there is a single response for each request, and that requests and servlets have URL-like pathnames. None of that is true for SIP.

Amongst others, steplets add the following two main features that are neither provided by SIP Servlets nor by the Java Servlet Specification:

1. Dynamic steplet list. With SIP Servlets, as with general Java servlets, when a message arrives, the servlet engine consults a static configuration descriptor to determine the servlet that will handle that message. The Service Broker, however, determines an initial steplet list for a message by looking up the “To” (or Request URI) or “From” (or P-Asserted Identity) address in a database (the HSS in an IMS network). Once a steplet is executing, it has the ability to append a different steplet to the Steplet List for that message. This ability to chain steplets during execution of the code enables dynamic determination of logic to be executed and/or applications to be invoked. This determination can be based on information obtained from outside the Service Broker and even from outside the IMS. It can be based on results from previous application invocations, or results from Presence, Location, and/or Policy Servers.

2. Steplet wait capability. The freedom to blend services that have an interactive element can give rise to long wait times, whether a steplet would need to wait for a call to be answered, for a response from a slow network database, or from an interaction with a user via an Instant Message and a link click. With SIP Servlets, which use the underlying Java wait/notify facilities, all of these would tie up threads, reducing the number of subscribers that can be handled by the system and, perhaps more importantly, limiting the types of blended services that are practical. In contrast, the steplet wait capability provided in the Service Broker relies on incoming SIP messages as the objects on which steplets wait. This allows steplets to release their threads while waiting.

The latter feature significantly improves efficiency. Without an explicit wait facility, a waiting servlet could tie up a thread many minutes. But if steplets use an explicit wait facility, it is reasonable to assume that a Service Broker steplet will tie up a thread for, say, at most 20 milliseconds. With an explicit wait facility, the Service Broker could function smoothly with a pool of, say, 30 worker threads, while without it, the Service Broker might need thousands of worker threads.

3. SERVICE EXAMPLES

We have implemented two applications using the previously described infrastructure. Therein, the Service Broker communicates with Lucent’s commercial IMS components via SIP and with IPTV components via SOAP/XML. The IPTV components include different commercial set-top boxes (such as Amino and Microsoft Media Center Edition) and a Linux-based IPTV server.

3.1 IPTV-Based Call Handling

The first example we present follows the blended IPTV/telephony application described briefly earlier in Section 1. This application integrates the handling of phone calls and the recording or pausing of TV content during phone calls. Figure 2 illustrates the roles of this application’s major components.

When Alice starts watching TV, her set-top box sends an event to the IPTV server, which forwards the event to the IPTV service
package (step 1). This event triggers the IPTV service package to store Alice’s status in the user data (step 2). Now, when Bob initiates a call to Alice, call event signals pass to Alice’s Serving Call Session Controller (S-CSCF). Because Alice has subscribed to the call handling application, the S-CSCF is configured to pass the call event signals to the Service Broker (step 3).

Meanwhile, Alice’s phone will start ringing. Now Alice can use her TV remote control to send the call to her answering service or to reject the call. When the set-top box receives either of these commands, it forwards the request to the IPTV server, which sends it to the IPTV service package. The service package sends a message to the Service Broker, which specifies that the answering service is receiving the call or that the call is rejected. In either case, the Service Broker ends the call to Alice’s phone.

If Alice answers the call with her phone, the S-CSCF gives the call receipt to the Service Broker (step 8). The Service Broker then forwards the SIP message through the S-CSCF to Bob’s phone (step 9), establishing the call. Meanwhile, the Service Broker sends a message to the IPTV service package, which instructs the IPTV server to remove the call notification information from Alice’s TV screen and to initiate temporary recording of the TV content (step 10). The IPTV server instructs a network DVR to record the content (step 11).

When Alice hangs up, the call termination event is passed through the S-CSCF to the Service Broker and to the IPTV service package, which informs the IPTV server that recording may end (step 12). The IPTV server alerts Alice that the recorded content is available (step 13), and Alice selects a playback option to view the material (step 14).

### 3.2 Remote Approval

In the previous application the Service Broker infrastructure mediated events from the IMS to the IPTV world. In contrast, we next present an example for the class of applications where our infrastructure enables the IPTV service to trigger events in and change state according to responses from the IMS world.

Content access control for TV services is a powerful feature, e.g., for preventing children from watching inappropriate content, spending an inappropriate amount of time watching TV, controlling the fees for pay-per-view programming, etc. Currently deployed content access control mechanisms are very limited in their functionality and convenience. Basically, they allow content authorization only from the content-requesting end point, which implies that for successful authorization the privileged user has to be physically present. Alternatively, the authorization information may be disclosed to the requesting user through an independent communication channel (e.g., telephone or IM). However, the latter approach is unreasonable and undermines the power of access control mechanisms.

A possible end-point solution is the use of remotely configurable set-top boxes (STBs). Nevertheless, the underlying model still assumes service provisioning to physical end points instead of subscribed users (independent of the used end devices). Hence, this approach requires all STBs to implement content access control features and to provide interfaces for remote configuration. In a home with multiple STBs the privileged user (mother, father) still needs to control and configure each single end device to maintain consistent access control rights. To overcome the aforementioned limitations, we leverage our Service Broker infrastructure to enable Remote Approval. Figure 3 depicts the infrastructure and the signaling flow for a Parental Control example scenario.

While parents Alice and Bob are at work, their children are at home supervised by Nanny. She has logged in to the IPTV service with a restricted user account set up for the children by Alice and Bob. When Nanny tries to access a TV programming that is...
are the following:

- Mapping identifiers among various component applications is a challenge. Mapping of telephone numbers or IM body names, user IPTV subscriber names, and IPTV server or set-top box addresses is required. These mappings are handled by the Service Broker IPTV service package. Separating a user id from a set-top box or TV id would require additional input from the users to establish their identities. This information could also be maintained in the IPTV service package, or in the IPTV middleware server.

- Scaling applications to large user communities is a common problem. In this or related applications, scaling may be limited by requirements for sharing information between the user set-top boxes and servers. For example, tracking what station a user is watching by monitoring all channel changes could overload the IPTV server.

- Blending applications may create interference on user devices. For example, in our first application, outputting caller info and associated command menus can interfere with displays from the TV content provider (under control from the IPTV server). In some cases, avoiding this interference might require some modification of the server or set-top box. Likewise, gathering input from the TV viewer without interference from the IPTV server might require service coordination. Similarly, devices suitable for one part of a blended service might not be readily suitable for other parts. For example, gathering user input for phone controls from a TV remote control requires careful control design.

4. RELATED WORK

Blending IPTV and telephony applications requires integration of the control and media processing functions of two distinct forms of communication. IPTV applications focus on user transactions involving services obtained through longer-term sessions. Telephony applications focus on user transactions involving real-time interactions associated with shorter-term sessions. While blended IPTV/telephony applications are still novel, substantial work has addressed the combination of Web and telephony functions.

IMS specifies network interfaces and collections of servers to handle telephony events and to engage application servers that may implement Web-based services [1]. IMS specifies both SIP and Parlay interfaces [5] between call handling components and the application servers. However, it is out of scope of IMS standards to specify interactions or relationships among application servers. Consequently, the specifications neither include service bundling and blending nor specifically address IPTV applications.

Service Delivery Platforms, including Connected Services Framework from Microsoft [6] and Websphere from IBM [7] are software suites that support the interaction of Web-based applications and telephony events. These software packages also address application interactions. However, these packages do not themselves handle telephony events and they do not integrate real-time telephony events with other applications.

5. SUMMARY

We have presented and described an infrastructure that enables efficient and true blending of IPTV services. While based on the IMS foundation, our infrastructure introduces a novel component, the Service Broker. Utilizing dynamically loaded steplets, the Service Broker enables service interworking with unified control over multiple applications.

We implemented two service examples that illustrate the possibilities for efficiently blending IPTV services through the use of our infrastructure. In the first example, events from the telephony world triggered events in and controlled the IPTV service. In the second example, the IPTV service triggered the establishment of interactive sessions in the telephony an IM world. The implementation of these two examples revealed major challenges inherent to service blending efforts. For example, during the development we had to address scalability issues in
terms of signaling messages and state maintenance, and interworking issues in terms of user and address mappings.

6. ACKNOWLEDGEMENTS

We would like to thank Bill Roome for his leadership in developing the Service Broker architecture, and Kristin Kocan and her team for bringing the system to life.

7. REFERENCES


