♦ Session Initiation Protocol Services Architecture

Janet R. Dianda, Vijay K. Gurbani, and Mark H. Jones

The session initiation protocol (SIP) is a unifying protocol for providing integrated telephony and Internet types of services, such as Web, presence, instant messaging, and chat. To provide for the integration of these disparate types of services, considerations must be given from a network point of view. However, little attention has been given in the literature to identify the network elements required to provide these services or the mechanisms for integrating these different types of services for end users. This paper describes a network-level services architecture for SIP, including network functions and entities needed to support the services integration. We will discuss how services can be incorporated at different levels in the network, and the types of services typically created at each of these levels. We will also describe a service access and mediation function, which blends disparate types of services in creating a seamless and rewarding user experience.

Introduction

The session initiation protocol (SIP) is a standard protocol for enabling the integration of telephony and Internet services in a converged wireline, wireless, and Internet network. The basic SIP protocol is specified as a signaling protocol for establishing, modifying, and terminating multimedia sessions, ranging from multimedia conferences to simple point-to-point voice calls [10]. Since SIP itself is based heavily on the Internet Web and e-mail protocols [6, 7], it is easily extensible to provide Internet-type services such as Web, e-mail, chat, instant messaging, and presence. The registration capability of SIP is a natural vehicle for providing mobility and location functions. Furthermore, the SIP protocol and its extensions are being used in the network at different levels to support a variety of functions. Some examples are given below:

- I*nterface to SIP-enabled endpoints,* such as SIP hard phones, SIP softphones, personal data assistants (PDAs), and wireless phones.
- *Call/session control,* for example, interface to media gateway controllers, softswitches, proxies.
- *Media service control,* for example, interface to media gateways and media servers.
- *Service control,* for example, interface to application servers and services mediation functions.
- *Service creation/authoring,* for example, support of call processing language (CPL), common gateway interface (CGI), and servlets [16, 26].
- *Intelligent network/Internet protocol (IN/IP) internetworking,* for example, the SIP-based Services in the PSTN/IN Requesting Internet Services

Bell Labs Technical Journal 7(1), 3–23 (2002) © 2002 Lucent Technologies Inc. Published by Wiley Periodicals, Inc. Published online in Wiley InterScience (www.interscience.wiley.com). • DOI: 10.1002/bltj.14

Panel 1. Abbreviations, Acronyms, and Terms

3GPP—Third-Generation Partnership Project
AINAP—advanced intelligent network application protocol
API—application programming interface
BCSM—basic call state model
CGI—common gateway interface
CORBA*—Common Object Request Broker Architecture
CPL—call processing language
FSM—finite state machine
HTTP—hypertext transfer protocol
ICW—Internet call waiting
IETF—Internet Engineering Task Force
IM—instant messaging
IN—intelligent network
INAP—intelligent network application protocol
IP—Internet protocol
ISDN—integrated services digital network
ITU-T—International Telecommunications Union, Telecommunications Standardization Sector
JAIN*—Java* APIs for Integrated Networks
JCC—JAIN call control
JTAPI—Java Telephony API
MGCP—media gateway control protocol
OAM&P—operations, administration, maintenance, and provisioning
OSA—open services architecture
PC—personal computer
PDA—personal data assistant
PINT—PSTN/Internet Internetworking
PSTN—public switched telephone network
SAMM—service access, mediation, and management
SCF—service control function
SCP—service control point
SDP—session description protocol
SIM—SIP/IN Internetworking
SIMPLE—SIP for instant messaging and presence leveraging
SIN—SIP/IN Internetworking
SIP—session initiation protocol
SIPPING—session initiation protocol project investigation
SLEE—service logic execution environment
SMTP—simple mail transport protocol
SN—service node
SPIRITS—Services in the PSTN/IN Requesting Internet Services
SS7—Signaling System 7
SSP—service switching point
TCAP—transaction capability application part
UAS—user agent server
URL—uniform resource locator
VoIP—voice over IP

(SPIRITS)/PSTN/Internet Internetworking (PINT) standards [11].

- *General event subscription/notification mechanism,* being standardized in the Internet Engineering Task Force (IETF) SIP working group [13].
- *SIP-enabled instant messaging and presence,* being standardized in the IETF SIP for instant messaging and presence leveraging (SIMPLE) working group [12].
- *Wireless multimedia protocol,* being standardized in the Third-Generation Partnership Project (3GPP) [27].

To provide for the integration of these disparate types of services using SIP as the unifying protocol, we must consider the service architecture from an end-to-end network point of view, specifically, identifying the network elements required to provide these services and the mechanisms for integrating these different types of services for end users. **Figure 1** describes a functional services architecture at network level, including network functions and entities needed to support the services integration. This paper will discuss how services may be incorporated at different levels in the network and the types of services typically created in each of these levels. Then we will describe a service "blending" mechanism via the service access, mediation, and management (SAMM) function, which integrates disparate types of services in creating a seamless and rewarding experience for the subscribers.

Intelligence Everywhere

A user, deploying a SIP-enabled endpoint, has many options as to where he or she can access serv-

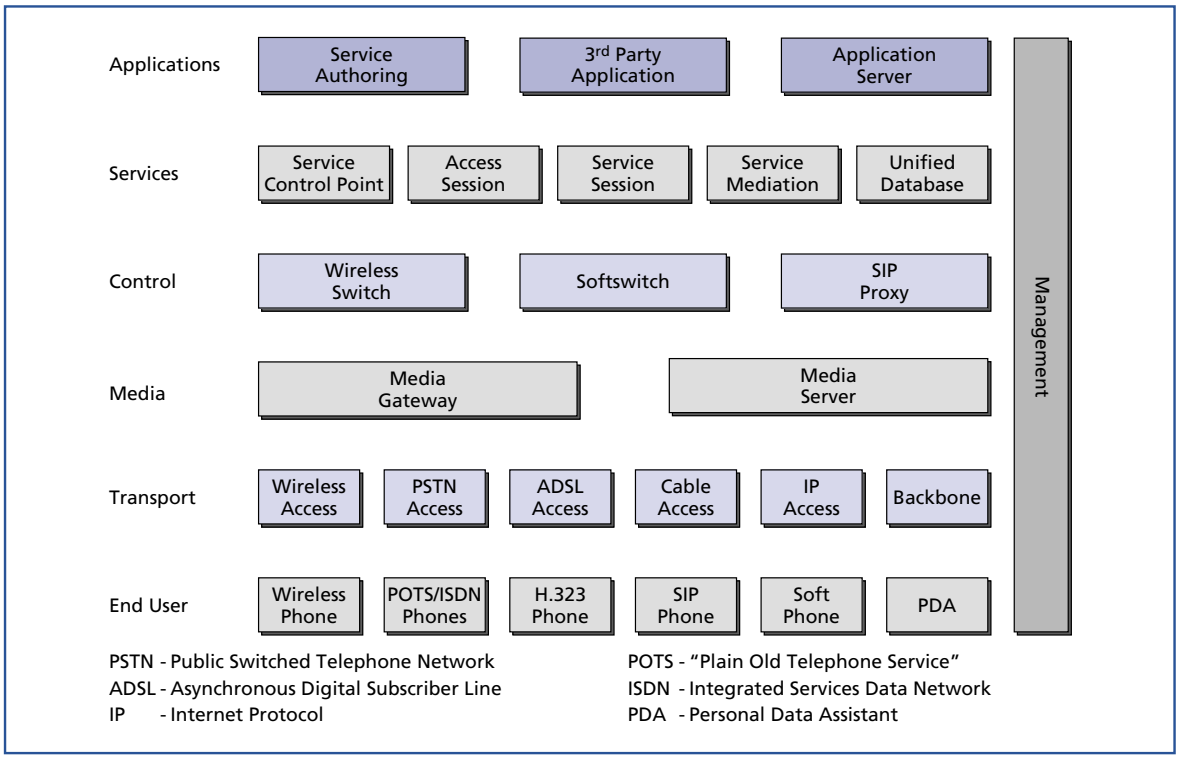

Figure 1.
Functional entities for a converged services network.

ices. The placement of services in a converged services network is dependent on the requirements of the service and the needs of the user. Services in different parts of a converged services network have different capabilities and needs, which must be addressed when determining the best place to implement and deploy a service.

The user may access services from a SIP-enabled endpoint that are developed and deployed in a variety of ways:

- SIP CPL, CGI or servlets,
- Parlay [18]/open services architecture (OSA), or JAIN2* [25], and
- Advanced intelligent network application protocol (AINAP), intelligent network application protocol (INAP), PINT, and SPIRITS services.

SIP service logic may reside on a variety of enterprise and network elements, including endpoints, SIP proxies, application servers, Parlay/OSA gateways, service control points (SCPs), and softswitches. The debate over services in the endpoint versus services in the network may be missing the point. Intelligent devices will have intelligence. Each device may make decisions based on the scope of service logic available to it. It can then signal requests to other devices using industry standards' protocols. Signaled devices may, in turn, make decisions based on the scope of service logic available to them. Some services will fit well on end devices; others will require the scope of distribution, reliability, and availability across a wide network.

For example, services that relate directly to the end user manipulating a particular endpoint sit well on an end device. These services might include multimedia-enabled versions of the services provided on wireless phones or integrated services digital network (ISDN) handsets, such as re-dial, speed call lists, hold, custom alert patterns and preferences

("ringing" or "instant messaging pop-ups"). Services provided to the user as an entity that spans multiple endpoints may be best provided on a network device that can be accessed by any endpoint. These services might include multimedia, multiparty conference calls, involving multiple endpoints for the user in a single call, or allowing a user to access personal services regardless of whether the user is using a wireless handset on the road, an analog phone in a hotel, or a personal computer (PC) in the office. Many services could reside on a local endpoint, such as calendars, address books, call screening, and speed call lists. However, since a user may have multiple endpoints (and endpoints can crash!), it is annoying to have to reprogram each endpoint with the same information. It would be nice to have a network service that could upload and store personal service information, and make it available to the user on any endpoint. Authorization and authentication of end users can be performed in the network, so that end users can communicate with each other with confidence that they know whom the far party really is. Otherwise, each endpoint would have to have the ability to detect spoofers itself.

The following subsections will describe the most common places services accessible from a SIP-enabled endpoint will be deployed and contrast the advantages and disadvantages of deploying services in the given area.

SIP-Enabled Endpoint Services

Intelligent endpoints are not a new concept. The use of intelligent endpoints to provide services has had numerous implementations. Many existing ISDN, wireless, and analog phones are available today that have embedded services. Most of these services can be categorized as data storage and retrieval. The ability to store frequently used phone numbers or to recall previously received phone numbers and names is integrated into a majority of the phones sold. Some wireless and wireline phones even have the ability to access and retrieve e-mail and Web content. What all of these phones have not had the ability to do until recently was to have control over the actual call. In the past the phone acted as little more than a dumb call peripheral with little understanding of the state of the connections on a call. Any connection service required the phone to request the service from the network via keying actions with little feedback to the endpoint other than audio tones to inform the user or endpoint of the status or state of the call. The request was sent to the network and processed there, if the user had subscribed and paid for the service. Only services offered by the local exchange carrier were easily available. Access to new services had to wait until the services were offered by the local exchange carrier or on occasion could be accessed in a degraded fashion through clumsy access numbers and keying actions from a third-party vendor.

SIP changes everything by no longer requiring the use of the local exchange carrier for access to all services. From a SIP-enabled endpoint, a user can directly request a connection to another endpoint, to modify an existing connection, or to access directly a service anywhere on the network. All of this is done through direct communication with the other endpoint or via data network proxies. There is no longer a need to have a centralized switch to handle the call or provide services. Because SIP is a peer-to-peer protocol, an endpoint can have considerable understanding of the current state of a call and use this state to provide many services at the endpoint.

Many of the features currently implemented on centralized telephony switches such as hold, transfer, and call screening are easily implemented at a SIP-enabled endpoint. Because they are resident in the endpoint, there is no long-term subscription cost to provide the service. The user is free to select the services they wish to have resident and active on their endpoint. These services can either be embedded in the endpoint firmware or downloaded from the network upon user request.

SIP-enabled endpoints are available in many different varieties with capabilities and programmability varying considerably depending on the endpoint.

Hard SIP-enabled endpoints. Hard SIP-enabled endpoints are typically always powered up and network connected, which allows for continuous services. Because of dedicated operation and hardware assisted

voice processing, voice quality is typically superior to other types of endpoints.

- *Traditional hard SIP-enabled endpoints.* Many of the first-generation SIP hard endpoints are little more than existing ISDN and analog phones modified to use the SIP protocol with no new service capabilities. SIP service enhancements and programmability are limited or non-existent in many of these phones.
- *Advanced hard SIP-enabled endpoints.* Many vendors are now supplying SIP hard phones with advanced SIP capabilities. These capabilities typically include advanced programmability, integration with the Web, and large user-friendly screen displays.

SIP soft endpoints. Softphones resident on PCs have the most rich programmability capabilities with the ability to take advantage of all of the PCs resources, that is, nonvolatile storage, rich graphical interface, and multiple user input devices. Because these endpoints can be powered down, service interruptions are possible. Because these SIP-enabled endpoints run as applications on the PC, they must contend for resources with the other applications. In the past, voice quality and stability have adversely affected voice quality for software-based SIP-enabled endpoints. Microsoft's* recent developments in integrating SIP into the Windows XP* operating system have not only provided end users with instant access to a quality SIP-enabled endpoint, but have also provided improved quality and a standard component object model based application programming interface (API) for developing endpoint services.

Wireless and mobile SIP-enabled endpoints. The Third-Generation Partnership Project (3GPP) is in the process of standardizing SIP as the protocol of choice in the wireless infrastructure. Devices included in this area include the next generation Web-connected wireless phones with enhanced programmability and wireless network-connected PDAs. In the ideal conditions, voice quality in these types of endpoints is typically on par with dedicated hard endpoints. Because these types of devices can easily be powered down, the ability to provide continuous services from them is affected.

Network appliances. This type of SIP-enabled endpoint is not a general purpose SIP-enabled endpoint, but is typically developed to provide a specific service. These devices are typically developed as custom standalone hardware devices. Very little programmability is provided with this device with ease of installation and use taking precedence. Typical examples include weather monitors, reminder services, and instant messaging.

The use of intelligent SIP-enabled endpoints opens the capability for end users to easily configure the services implemented on the endpoint either through a graphical user friendly interface on the SIP-enabled endpoint or through Web browser access. Most new SIP-enabled endpoints, whether implemented on custom hardware or implemented as software on a PC or PC-like device, provide for a much richer user interface than a traditional telephony endpoint. This allows the user to easily implement complex services and service interactions with little user effort required. For example, many central office switches currently provide call screening services, which either restrict or allow calls to a given end user. End-user programmability of these services is either through complex keying actions or confusing multilevel voice menu systems. Because of the complexity of configuring these types of services, many users either do not subscribe to these types of service or do not fully take advantage of their capabilities.

SIP Network Services

Although it is possible to implement a large number of the existing class 5 features at the endpoint, it does not always make sense. Services such as hold, transfer, caller ID, and call screening make sense to implement at the endpoint because they are services that are user action oriented. They are invoked in response to a user action, or are setup specifically to a given phone.

Services can be accessed in a SIP network in several ways.

1. By requesting from a SIP-enabled endpoint to be connected to a specific server that provides a service,

2. By having a SIP proxy intercept an incoming or outgoing request and rerouting it to an application server, and
3. Requesting through standard Internet data protocols (hypertext transfer protocol [HTTP], simple mail transport protocol [SMTP]) for a server to create, via third-party call control, a SIP call, and optionally provide addition services.

SIP Network Components

In a typical SIP network, proxies are configured in a hierarchical architecture in order to provide SIP-converged services. A SIP proxy is much like a Web proxy in that its responsibility is to proxy a request to the requested destination. Because all requests into or out of a particular segment of the network must traverse the proxy server, it has the ability to provide a number of useful services. These services can be grouped as follows:

- *Security:* A proxy can provide security services at the edge of a network. To do this, it must work closely with a firewall to provide SIP access to the network both at the signaling and bearer level. Work is currently being done in the IETF to standardize this mechanism.
- *Accounting:* A proxy can provide an access point for network-level services related to authentication, authorization, and accounting.
- *Routing:* A proxy can provide a valued-added routing service based on user preferences or net-work activity and resources. This allows a SIP network to recreate traditional telephony services, for example, toll tandem, 800 service.
- *Application Access:* A proxy can act as an intercept or final destination for accessing applications servers.

Because SIP is heavily based on Web technologies, it is easy to develop and integrate converged services based on SIP and other Web technologies, such as SMTP, HTTP, and instant messaging (IM)/presence. For example, providing a multiparty conference services could involve the following scenario:

1. A user wishes to conference with several other people on a problem.
2. From a Web browser, the user specifies all of the call participants through a Web page form.
3. The Web server sends the request to the scheduling server.
4. The scheduling server subscribes to the presence server to be notified of all conference participants' status.
5. The scheduling server is informed when the presence of a call participant changes.
6. When all call participants are available, the scheduling server notifies the original requester that the conference can now take place and requests confirmation.
7. The person requesting the conference confirms.
8. The scheduling server through third-party call control calls each participant and connects them to a conference server.

The conference server bridges the connection and the call begins.

Candidate Network Services via SIP

There are several classes of services that are not good candidates for implementation on an endpoint.

Multi-endpoint interaction services. Most users have multiple phones and require services that are designed to terminate incoming connections to a specific endpoint based on specific criteria such as user presence, time-of-day, or caller. Terminating a call to a specific phone and then providing the service, which may include a transfer, is not as efficient or practical as providing the service in the network.

Continuous services. SIP-enabled endpoints, such as wireless phones or PC soft phones are not guaranteed to be active at all times. Services resident on a SIP-enabled endpoint will only work properly if the endpoint is powered up and actively connected to the network. For example, a call forwarding service embedded in a phone will not be able to operate on incoming calls if the phone is not powered up and connected to the network.

High resource services. Many services have resource requirements, which exceed the capabilities of the endpoint. For example, providing enough processing power at all endpoints in order to allow a complex multiparty conference or voice recognition is not practical or an efficient use of resources.

Service configuration. Even with more user-friendly interfaces, significant effort can be involved in config-

uring a soft endpoint to provide all of the services and service interactions required by a specific user. Different phones will typically have different input capabilities and have varying degrees of programmability. End users with multiple phones, who wish to provide consistent services on each phone, will be required to program each phone individually with the same configuration, for example, adding a new restricted number.

Duplication of services. SIP-enabled endpoints will typically have widely varying capabilities and degrees of programmability. Typical service providers will not want to implement the same service multiple times for each of the endpoints. By putting the service in the network, a service developer is able to provide the service to multiple heterogeneous endpoints using the common SIP protocol.

Service interaction and mediation. As more services become available for use on endpoints, interactions will inevitably occur and limit a user's ability to pick and choose service from multiple sources. These interactions can become very complex and often require significant understanding of the internals of a service along with telephony and networking concepts. Unless a vendor has provided the ability to interact with other services, interactions may be impossible. Expecting an untrained end user to be able to install and configure multi-vendor services is probably not feasible at this time. Interaction and mediation frameworks are being designed to provide multi-vendor interoperability, but they are currently in their infancy, and currently only available at the network server level. A candidate framework for this purpose is described in "Service Session, Access, Mediation, and Management for an SIP Services Architecture" below.

Microsoft's recent integration of a SIP-enabled endpoint into every new version of its operating system has solved the current problem of getting quality and usable SIP-enabled endpoints distributed and available on the desktop. Demand now is moving toward building services at the endpoints, deploying a SIP network infrastructure, and developing SIP application servers, which provide services in the network.

SIP Telephony Integration

Internetworking of the public switched telephone network (PSTN) and Internet is of extreme importance for services. Instead of replicating PSTN-like services in the Internet, it is desirable for the Internet to leverage as many services as possible from the legacy PSTN. There are three efforts within the IETF to realize hybrid services: PINT, SPIRITS, and SIP/IN Internetworking (SIN). PINT and SPIRITS are IETF working groups and SIN is an IETF design team. The working group for PINT is currently dissolved, having met its deliverables. SPIRITS and SIN are currently active in the IETF. This section presents an overview of PINT, SPIRITS, and SIN from a service perspective. Underlying PINT, SPIRITS, and SIN is the use of the IN to realize hybrid services. IN has already been established as the key technology of choice for providing such hybrid services [15]. PINT, SPIRITS, and SIN focus on the building blocks from which a broad range of IN services can be constructed. The remaining of this discussion will assume that the reader is familiar with the IN and its accompanying protocols (INAP, transaction capability application part [TCAP]) and network entities (SCPs, service nodes [SNs], IP). Uninitiated readers are referred to [4] for the IN standard and [14] and [15] for the application of IN in converged networks.

PINT

PINT is designed to combine Internet applications and PSTN telecommunication services in a way that enables Internet applications to request PSTN telecommunication services. The Internet is used for signaling interactions, while the media portions are carried entirely over the PSTN. The service aspects of PINT as they apply to the PSTN are discussed in detail in [14]. Here we simply discuss the PINT architecture and the benchmark PINT services for the sake of completeness.

The PINT functional architecture is depicted in **Figure 2.** It is worth noting that PINT chose SIP as the enabling protocol, and in fact PINT defined extensions to SIP and session description protocol (SDP) [19] to name and describe the converged services. PINT clients and servers are SIP clients and servers. SIP is used to carry the request from the PINT client over the IP network to the correct PINT server. A PINT system uses the SIP entities, such as proxy and redirect servers for their usual purpose, but at some point,

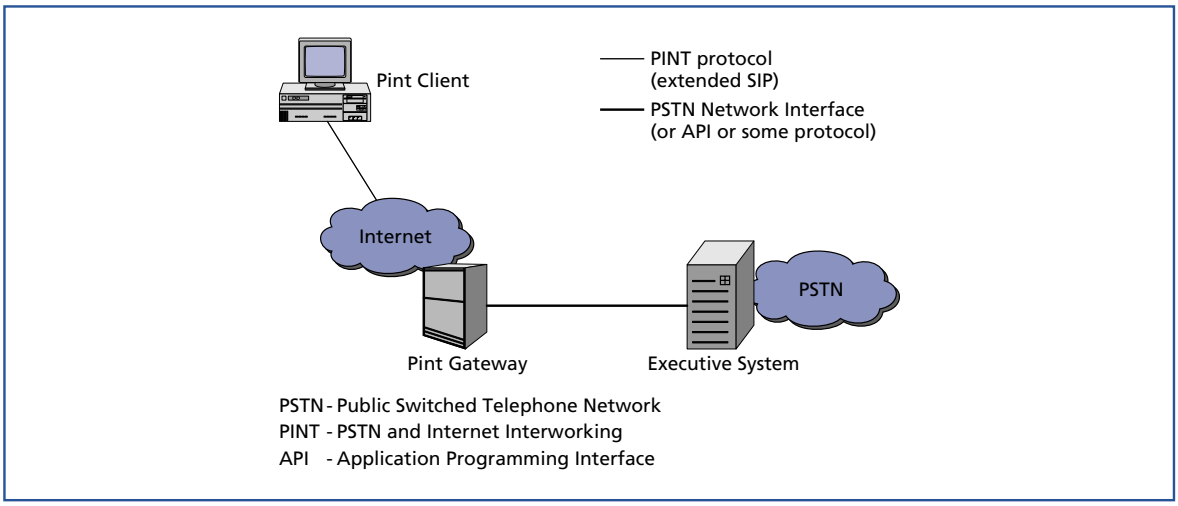

Figure 2.
PINT services architecture.

there must be a PINT server with the means to relay received requests into a telephone system and to receive acknowledgment of these relayed requests. A PINT server with this capability is called a PINT gateway. Such a gateway will terminate the PINT signaling and interface with the PSTN to provide a set of telephone network services. The PINT gateway may be directly connected to the PSTN through a telephone network interface, or it may be connected through some other protocol or API to an executive system, which in turn is capable of invoking services within the PSTN. An executive system could be an IN component, such as an SCP or an IP private branch exchange system.

A PINT gateway and the executive system with which the gateway is associated exist to provide services to PINT requestors. A PINT requestor initiates a request from the Internet, using a PINT client. One of the benchmark PINT services is click-to-dial-back. With this service, a user can make a request through an IP host to establish a PSTN call with another party. The requestor of the PINT service must have voice access to the PSTN (via the telephone) and Internet access (via a PC). The called party must be on the PSTN and need not have Internet connectivity. A typical example of this service is on-line shopping, where a user, browsing through an on-line catalog, needs to talk to a customer sales representative. The user, acting as the PINT requestor, clicks on a button, thus inviting a call from a sales representative. As in all-PSTN toll-free service, flexible billing arrangements can be made on behalf of the service provider. In addition, the PSTN can route the call, depending on factors, such as the time of day, day of week, and availability of agents in different locations.

SPIRITS

SPIRITS is a mirror image of PINT. Whereas in the latter the requests for service initiate in the Internet domain and terminate in the PSTN domain, SPIRITS concerns itself with requests for services that originate in the PSTN domain and necessitate a termination in the Internet domain. SPIRITS is an IETF working group that addresses how services supported by IP network entities can be started from IN requests, as well as the protocol arrangements through which PSTN can request actions to be carried out in the IP network in response to events (IN triggers) occurring within the PSTN/IN.

The SPIRITS architecture consists of three potentially independent entities: the SPIRITS client, the SPIRITS server, and the PSTN/IN requesting system. The SPIRITS server resides in the Internet domain, and receives service notifications from the PSTN/IN and sends optional responses back. The SPIRITS client

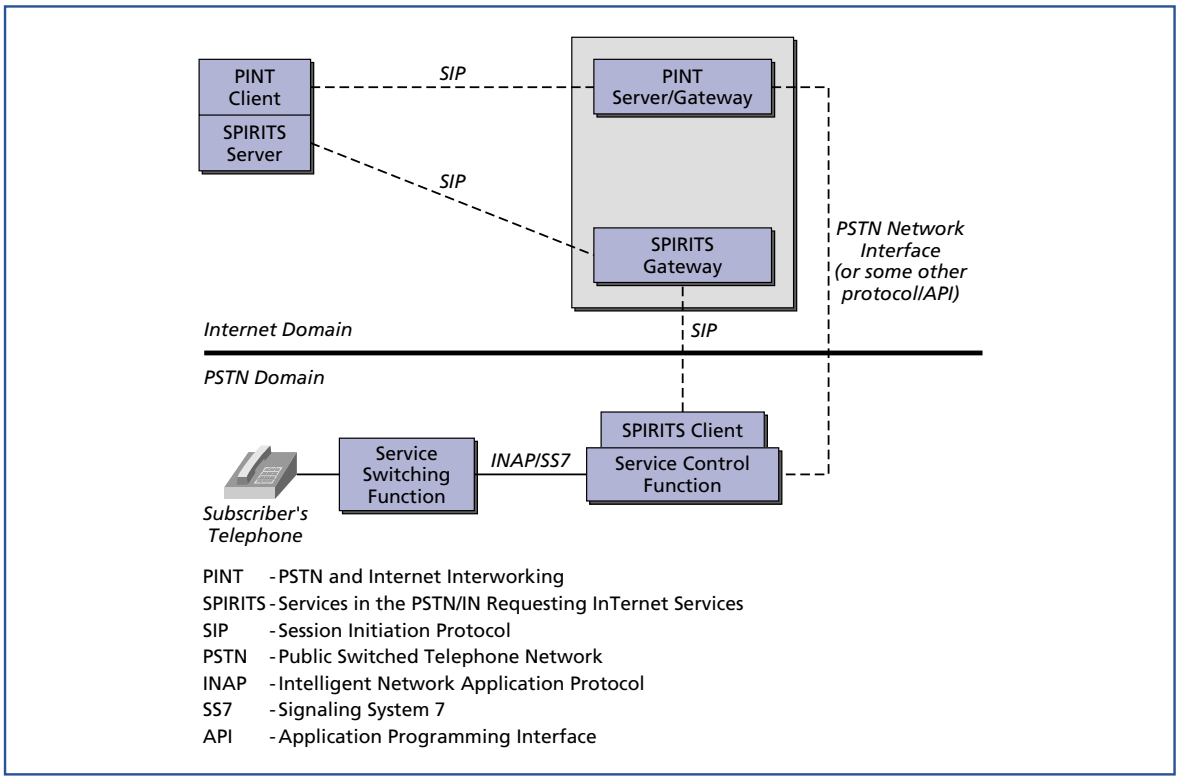

Figure 3.
SPIRITS architecture.

is located in the PSTN domain and initiates a request to the SPIRITS server for some action to be performed in the Internet domain. The PSTN/IN requesting system is realized by a SPIRITS gateway, which serves as an intermediary between the SPIRITS server and the SPIRITS client. The overall SPIRITS architecture is depicted in **Figure 3** [24].

Figure 3 includes many other entities in the SPIRITS architecture, besides the three mentioned above. It also shows a PINT client and a PINT server/gateway. While PINT is not absolutely essential for SPIRITS services, it does serve two critical functions in the architecture, namely, it allows for a SPIRITS host to register itself, and allows the SPIRITS host to avail itself of the SUBSCRIBE/NOTIFY PINT extensions for service subscription. Entities in the PSTN domain include the service switching point (SSP) and a service control function (SCF). An SSP is a switch in the telephone network, and an SCF is an adjunct that executes the service logic for a phone call. During the execution of the call logic, if certain criteria are met, the SPIRITS client, which may be co-located with the SCF, notifies the SPIRITS gateway of the events that need to be propagated to the SPIRITS server in the Internet domain. In this manner, PSTN events are exported to the Internet domain, and enable services to be executed there.

One of the benchmark SPIRITS services is Internet call waiting (ICW) [17]. ICW enables a subscriber engaged in an Internet dial-up session to be notified of an incoming call to his/her phone line. The notification may result in the subscriber tearing down his/her Internet session to accept the call, or it may result in accepting the call using voice over IP (VoIP), or forwarding it to some other number (a cell phone, or a second home line, for instance). Following the SPIRITS model, the notification of ICW originates in the PSTN domain. The SCF recognizes that the called

subscriber is currently on-line. It subsequently requests the SPIRITS client to notify the SPIRITS server of the pending call, and waits for the disposition of the call from the subscriber. The service to be invoked, in this case, displaying a pop-up dialog on the subscriber's IP host, occurs in the IP domain.

Based on the success of extending SIP for PINT and the use of SIP in pre-SPIRITS implementations [17], SIP has been chosen as the transport protocol for SPIRITS [5].

SIP/IN Internetworking (SIN)

As SIP becomes an important protocol for IP telephony, there is a growing need to support existing IN-based applications in a SIP-based IP telephony environment, particularly for IP host to PSTN phone calls. In the fall of 2000, the IETF formed a design team called SIN to handle this work. Specifically, SIN was chartered to solve the problem of accessing IN services residing in the PSTN domain from IP endpoints running SIP. The benchmark SIN service is Internet-originated 800 PC-to-phone calls, possibly involving other PSTN/IN entities, such as voice recognition peripheral and media servers for playing announcements and collecting digits.

In order to discuss the work being done in SIN, it is necessary to outline how IN services are accessed from PSTN. An SSP processing a call can temporarily suspend the call processing and request instructions from a service logic entity, called an SCP, on how to further proceed with the call. The points where a call can be interrupted are standardized within the basic call state model (BCSM). The BCSM model contains two processes, one each for the originating and terminating part of a call. When the SCP gets an request for instructions, it can reply with a single response, such as a simple number translation augmented by criteria, such as time of day or day of week, or, in turn, get into a complex dialog with the switch. The situation is further complicated by the necessity to engage other specialized devices, which collect digits, play recorded announcements, and perform text-to-speech or speech-to-text conversion. The related protocol, as well as the BCSM, is standardized by the International Telecommunications Union, Telecommunications Standardization Sector (ITU-T), and known as the INAP.

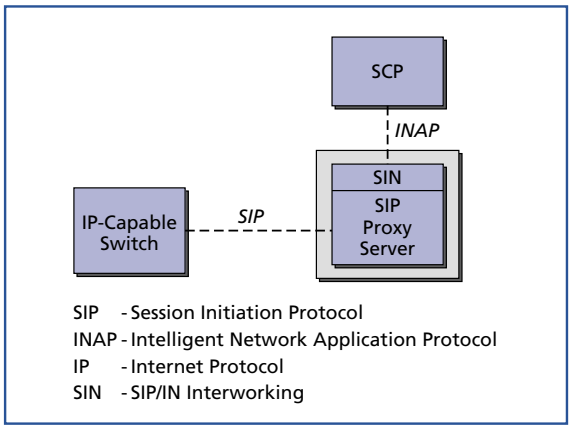

Figure 4.
SIN architecture.

The SIN architecture is described in **Figure 4** [22]. The architecture depicts a SIN-enabled SIP proxy that interacts with the SCP to provide services to the SIP-enabled endpoints. In case of the 800 translation service, the proxy can interact with the SCP through the SIN layer, translate the 800 number to a valid routing number, and forward the call to an Internet telephony switch for delivery to the PSTN. Note that the IP switch will straddle the PSTN and Internet domains.

Three levels of internetworking come into play in SIN: mapping the SIP protocol state machine to the IN BCSM call model, INAP/SIP message sequence mapping, and INAP/SIP parameter translation. The mapping of call models between SIP and IN [8, 9] demonstrates the services that are possible within the framework of SIN. Current work in SIN is focused on the INAP/SIP message sequence mapping and INAP/SIP parameter translation.

SIP Internet Integration

A presence and instant messaging system allows users to subscribe to each other and be notified of changes in state, and for users to send each other short (usually text) instant messages [1]. Although IM and presence are often coupled together, they are in reality two separate applications. Coupling them together provides an integrated manner of communications, wherein presence implies the ability to send an instant message to the recipient (if the recipient is

present, the sender can send an instant message to the recipient. However, note that presence does not imply *availability;* a recipient may be present, but unavailable, that is, busy).

SIP is a natural protocol to leverage IM and presence. The mechanisms needed in an IM protocol are very similar to those required for establishment of an interactive session, that is rapid delivery of small content to a user at their current location, which may be changing as the user moves [20]. Likewise, a presence system can avail itself of the SIP infrastructure of registrars and location servers. After all, the presence of a SIP subscriber is already known to a registrar through the SIP REGISTER method [21].

SIP and IM

An IM service has two entities: senders and instant inboxes [1]. Senders construct and dispatch instant messages, and an instant inbox is a receptacle for an IM. Note that the recipient does not have to be present in order for an IM system to deliver an IM message. The IM will be stored in the recipient's instant inbox until he/she becomes available, whereupon it will be presented to him/her.

The MESSAGE extension enables IM in SIP [20]. When a user wishes to send an IM to another, the sender formats and issues a SIP MESSAGE request. The body of this request contains the message to be delivered. The request may traverse the SIP network, passing through SIP proxies to arrive at the destination user agent server (UAS). The proxies may fork MESSAGE requests, much as they do with INVITE requests. Provisional and final responses will be returned to the sender, as is the case with any other SIP request. Normally, a 200 OK final response will be generated by the UAS of the request's final recipient. The 200 OK implies that the IM was delivered at the final destination, not that the user has actually seen it.

SIP and Presence

The presence service has two distinct set of clients: presentities and watchers [1] (see **Figure 5**).

Presentities provide presence information to a presence service for storage and distribution. The presence service informs *watchers* of the change in presence information of presentity. Watchers are further divided into *fetchers* and *subscribers.* Fetchers simply request the current value of some presentity's presence information from a presence service. Subscribers, on the other hand, request notification from the presence service of future change in some presentity's presence information.

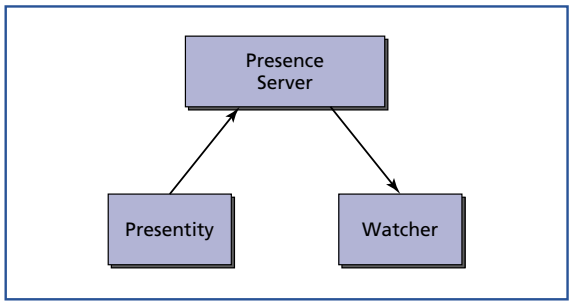

Figure 5.
Overview of presence service.

When a subscriber wishes to learn about the presence information of some user, it creates a SIP SUBSCRIBE request [21]. This request identifies the desired presentity in the request uniform resource identifier (URI). This request is routed through the SIP network and finally arrives at the presentity. The presentity authenticates and authorizes the request and sends a "202 Accepted" response to the subscriber. It also sends an immediate NOTIFY message to the subscriber, informing it of the present state of the presentity. As the state of the presentity changes over time, it generates NOTIFYs for all subscribers interested in that presentity. **Figure 6** depicts a simple time line diagram outlining the flow of messages between presentity and a watcher. Note that in Figure 6, the presence server and presentity are co-located. Nothing prevents them from being distinct SIP entities; the call flow depicted in the figure does not change.

Service Session Access, Mediation, and Management for a SIP Services Architecture

This section discusses a candidate framework for a SIP services architecture in a converged network, which enables multi-vendor service interoperability through service interaction mediation. It also provides service sessions that coordinate multimedia and multiple parties, service access, and service management.

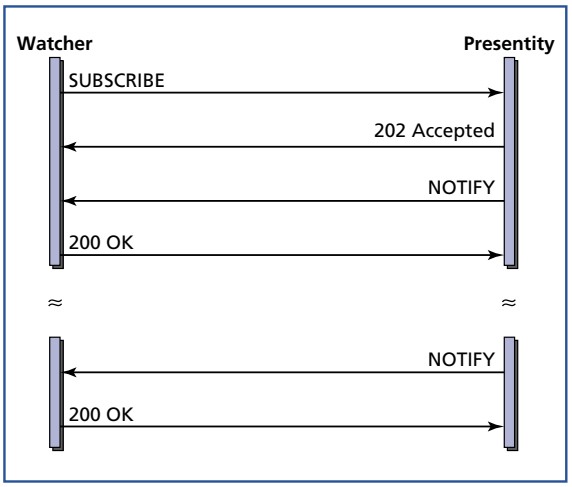

Figure 6.
SIP presence call flow.

The following terminology will be used in this section.

- *Application.* An application means software that provides a service or part of a service; it may reside locally or on a remote application server.
- *Service.* User visible behavior that adds value to the communication experience.
- *Service Blending.* Service blending enables multiple applications that provide services, written by multiple third parties, to register to act on the same event, for the same user, according to roles and precedence, in order to provide coherent service offerings to subscribers.
- *Service Mediation.* The runtime software mechanism that supports service blending.

SAMM Capabilities for SIP Services

SAMM provides a framework for service session access, mediation, and management for multimedia-enabled, next-generation services offered over IP networks to SIP-enabled endpoints (see **Figure 7**). SAMM service mediation enables services created by third parties, service providers, and equipment vendors to work together in a service logic execution environment (SLEE) [2] expanding on the service architecture concepts specified by JAIN SLEE. SAMM will support a third-party programmable service authoring environment that enables deployment of multimedia-enabled services, while resolving feature interactions and integration into existing operations, administration, maintenance, and provisioning (OAM&P) elements [3]. The applications will be written to industry standard APIs, including Parlay/OSA, JAIN, and SIP. Intelligent endpoints may access the services through multimedia-enabled access sessions via HTTP or through direct service signaling via a SIP proxy. When the user selects a network communications service, the access sessions or signaling from a proxy will start up service sessions software that provides a shared context for the services and the finite state machines (FSMs) that drive the session. SAMM is designed to interface by selection to any underlying call model through protocols, such as JAIN call control (JCC) and Java Telephony API (JTAPI). It may reside on SIP-enabled endpoints, SIP proxies, Parlay/OSA gateways, application servers, or softswitches.

Service Sessions

Service sessions manage the multimedia, multiparty context for a communications session, in order to facilitate intelligent service actions on behalf of the user, and track media resource usage by the user (see **Figure 8**).

A service session serves as the hub of a wheel, coordinating the components that together provide the service. At the application layer, a user agent component represents a user who can span multiple different endpoints (analog, softphones, and wireless phones) and take on many different personae (roles). A service session component is initiated when the first service signaling request is received through the access session. It invokes service mediation to enable multiple service logic components (applications) to respond to a service-affecting event. It invokes the appropriate endpoint agent components to communicate service information back to the user. The core service logic components that receive the events are designed to be endpoint and transport neutral. They are designed to provide any where, any media services to the user. It is therefore the endpoint agent's responsibility to translate the service logic components' results into endpoint specific behaviors, and to interact appropriately with the endpoints.

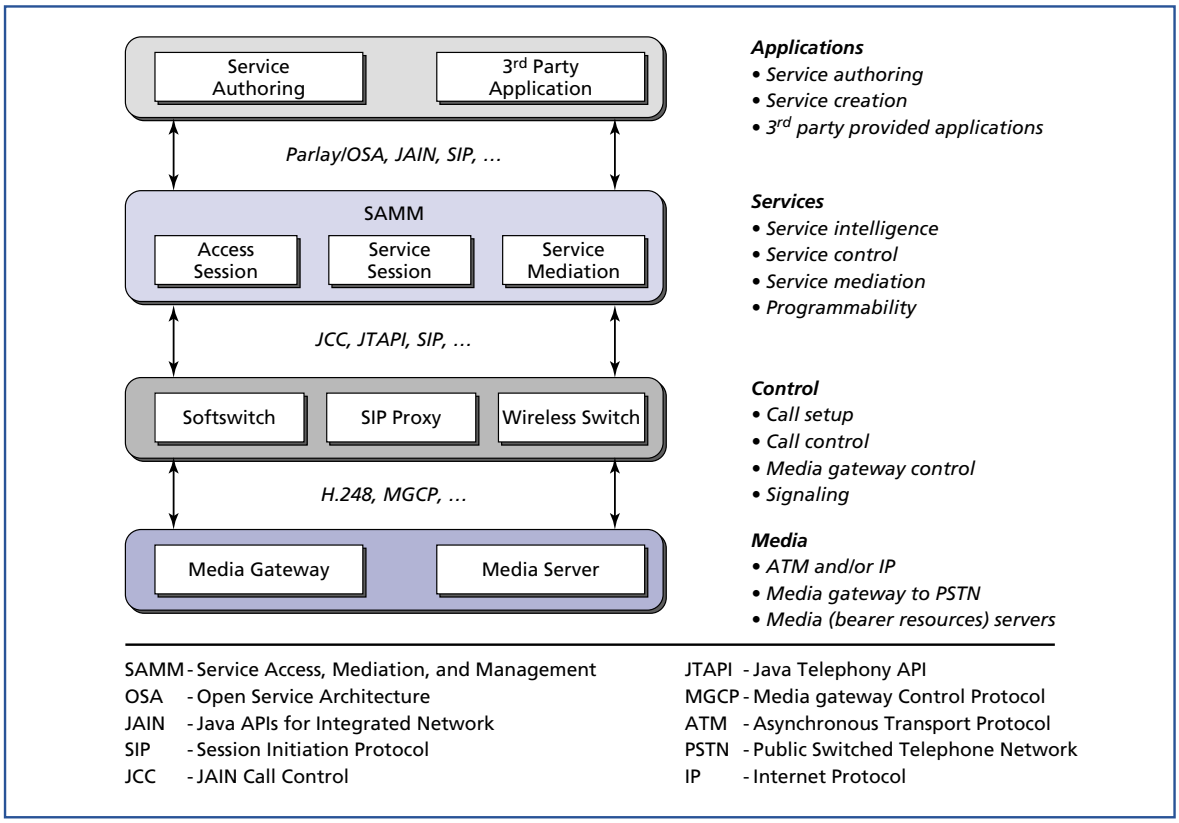

Figure 7.
SAMM functional entities and interfaces.

Possible endpoint directives may be transported as data from the service logic components to the endpoint agent. The endpoint agent then selects the relevant directives, given the actual endpoint's multimedia capabilities, or translates the directives to be compatible with the endpoint's multimedia capabilities. For example, the service logic components may convey to the endpoint agent, in a generic fashion, the need for a dialog with the end user. The endpoint agent customizes service presentation on the terminal based on user's current endpoint capabilities and preferences (voice only, voice/data, voice/data/video). The service session sends any resulting network requests to a connection manager component, which communicates with the call control/media gateway controller on a softswitch (if the communications session involves the PSTN), using protocols such as JCC or JTAPI, or it may communicate to a SIP proxy via SIP, for end-to-end SIP communications. The media gateway controller makes requests of a switch or gateway according to standard protocols such as H.248 or media gateway control protocol (MGCP). The service session may also send requests to resource manager components, which communicate with media servers using H.248 or MGCP. Responses from the switch, gateway, or media servers are sent back to the corresponding service session. Service sessions may reside on SIP proxies, application servers, or Parlay/OSA gateways.

Service Access

Service access receives service signaling from the user, authenticates the user, and invokes the appropriate service. Access sessions may be provided as user servlets that run on a Web server, and communicate with the service sessions using commu-

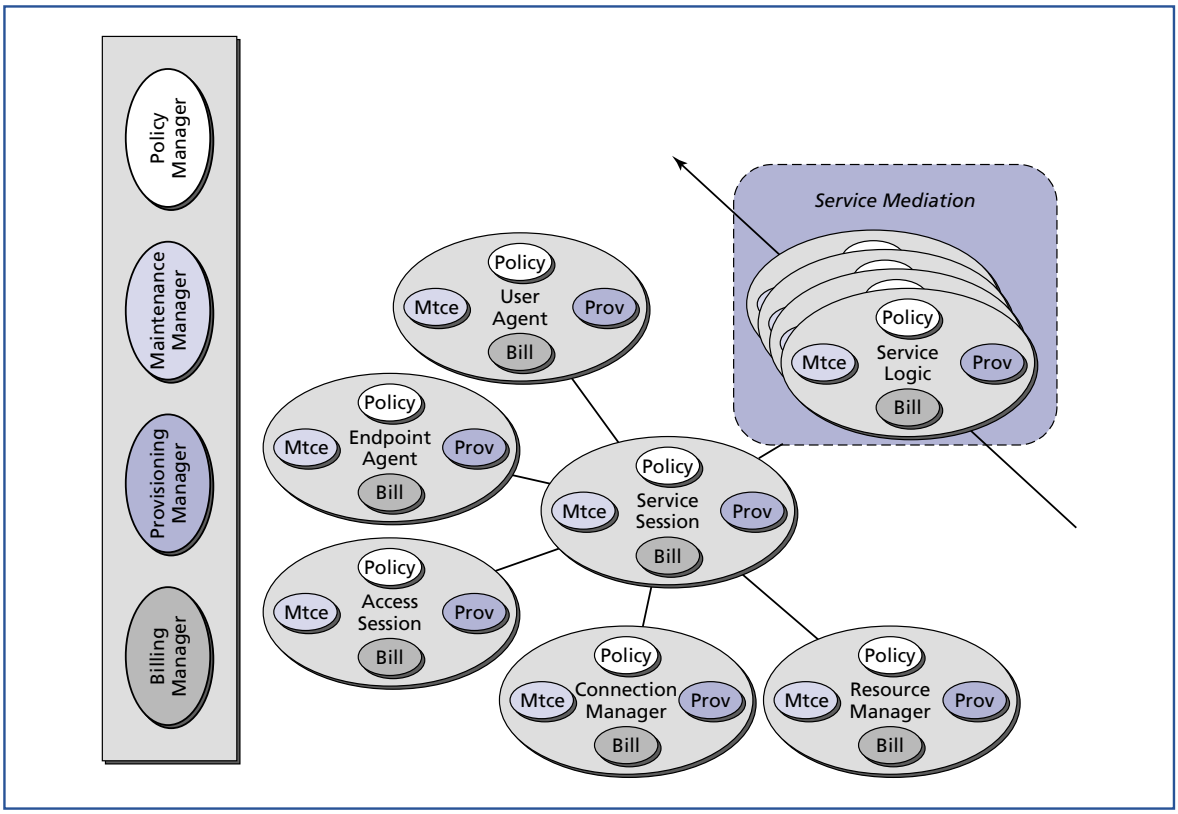

Figure 8.
The service session coordinates SAMM components to provide services.

nications mechanisms such as Java remote method invocation, simple object access protocol, or CORBA*. Access sessions provide "multimedia dial tone" for intelligent endpoints. For example, the subscriber may select services using touchtones, visual mechanisms, such as clicking on a Web page that translates the request into IP data, or by using a speech recognition system.

Service Signaling from SIP-Enabled Endpoints

SIP methods currently operate at the call/session control level. Service signaling is being considered by the session initiation protocol project investigation (SIPPING) working group [23]. This working group is chartered to document the use of SIP for several applications related to telephony and multimedia, and to develop requirements for any extensions to SIP needed for those applications. The challenge is determining how to unambiguously articulate services among endpoints across the network. Using the current SIP protocol, there is a limited vocabulary for articulating service behaviors. In traditional analog telephony, service signaling was done by entering touchtone codes in-band. The service provider who received the tones must then map them to a pre-determined service. This required agreements by multiple parties concerning how to signal the service.

Service Signaling Through Portals and Web Browsers

Service signaling could be done through portals to service servers, using visual or audio human language, through Web pages, speech recognition systems, or combinations of the two. Intelligent endpoints with Web browsers have the ability to do service signaling using HTTP. The Web browsers can present Web pages from various uniform resource lo-

cators (URLs). The user may select a service through clicking on a Web page, and/or through filling out a service description menu with scroll down bars for choosing options. The service selection is contextual and unambiguous, presented to the user through human language. It is not necessary to get agreement concerning special codes. Instead, the Web browser may invoke the appropriate services programmatically, through an API. The SIP protocol could then be used to actually set up the calls among multiple users, through third-party call control.

Service Mediation

SAMM enables service blending of independently developed third-party applications for a given subscriber. Service blending enables multiple applications (service logic components), written by multiple third parties, to register to act on the same service-affecting event, for the same user, according to roles and precedence. It provides service mediation for service-affecting events. Users will want to access new third-party services, written across multiple standards, as well as legacy services. It is relatively straightforward to interwork the first set of services offered or envisioned in a hard-coded, monolithic way. It becomes increasingly difficult, time-consuming, and expensive to integrate the nth service (which was not envisioned when the original set of services was created), with the $n - 1$ services that came before it, unless all the services adhere to a generic and extensible set of rules concerning how they will respond to service-affecting events.

Service Blending for SIP with Industry Standard APIs

SAMM service blending is designed to work with third-party applications written across a set of industry standards, and with underlying network protocols that allow use of network resources. It provides an open architecture that is extensible by the service providers. SAMM software is designed to act as a proxy between applications and call control. It does not change the industry standard application and call control APIs in proprietary ways. It just provides an additional set of plugs and adapters that allow applications written for different (or the same) APIs to be integrated together by the service integrator.

The service integrator is defined as a person or software entity that builds coherent service offerings, gathering together third-party applications, and specifying how those applications will plug into the service mediation capabilities. Third-party applications may be written without service mediation, and later be plugged in to work with service mediation, without modification to the applications themselves. The service integrator generates a thin client stub to send and receive messages from that application. Note that the service integrator is taking responsibility for the semantics of the service blending, to ensure that the resulting service offering makes sense. Over time, the service integrator may become automated and dynamic, as service descriptions and rules are adopted by industry standards. The service mediation capabilities of SAMM themselves can evolve, and will take advantage of the automated and dynamic algorithms as they become available. Knowing where to put an application in reference to other applications for handling events is a natural requirement, if you want the services to play well together on your behalf. In other words, it gives you the power to coordinate services rather than rewrite or duplicate services that are already available and written by someone else. The advantage of SAMM service mediation is that only the integrator must figure out how to combine the services, based on semantic needs. The third-party application developers do not have to know or understand the other parties' applications.

The SAMM components that provide service mediation are designed to be adaptable to be compatible with service mediation standards, as they continue to be defined by Parlay/OSA and JAIN SLEE. Standards that do not include explicit APIs for service mediation can be plugged in via adapters. Since different standards may develop different APIs for basically the same set of events, the SAMM adapters will map events between external APIs, and the common internal ones.

The Location of SAMM in the SIP Network

Service blending may occur independently in various network elements owned by different stakeholders, at various layers, since it is hidden behind standard APIs among the network elements.

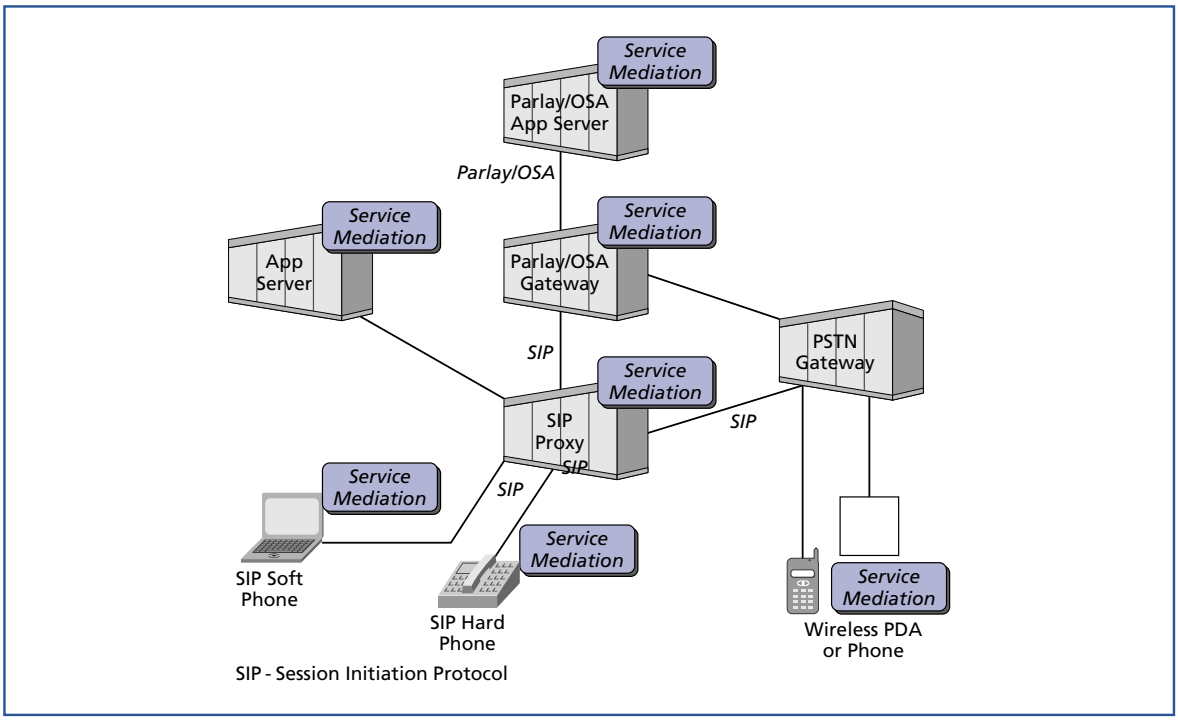

Figure 9.
Possible locations for service mediation in a SIP network.

SAMM could reside on a Parlay/OSA gateway. A Parlay/OSA gateway resides in the network, and receives Parlay/OSA requests from third parties. An application on an application server would communicate through its normal Parlay/OSA API to the Parlay/OSA adapter of SAMM, which plugs the application into service mediation. It would shield multiple applications from the lower level call control gateway. The service session could take resulting network requests, and convey them via SIP to a SIP proxy server. SAMM could reside on an application server to provide service mediation of events among multiple internal services, for example, and then make the resulting request through a Parlay/OSA API to a Parlay/OSA gateway. Refer to **Figure 9**.

SAMM could also provide service mediation directly on a SIP proxy server, or on an intelligent endpoint, such as a softphone, a Java-enabled wireless phone, or a PDA. It can provide a disciplined and generic approach to service mediation wherever it is needed, and it can provide service mediation on multiple network elements. As long as the network elements communicate via standard APIs, the service mediation occurring on one element is hidden from the service mediation occurring on another element. The key element is the semantics. The services must combine in ways that make sense to the subscriber.

Principles for Service Blending

Ordinary tools may detect for which events services are registered to respond—but not know how they should be combined to provide meaningful service offerings. SAMM provides development tools and a runtime environment to enable service integrators to combine the services together, to form coherent service offerings for subscribers, based on:

- *Preferences.* Understanding how the subscribers want their service to work (does the subscriber first want to forward a call, or try call waiting).
- *Roles.* Understanding what the services can do, and the context in which each service can sanely

execute; understanding the semantics of the services.

Applications written by equipment vendors, by service providers, and by third-party developers will all be integrated into service blending using the same service mediation mechanisms. This ensures total compatibility and extensibility among the applications, and avoids the possibility of equipment vendor applications somehow becoming incompatible with third-party applications.

The articulation of how the services may be combined to provide coherent and comprehensible service offerings embody the rules that drive the service sessions of the subscriber through its FSM, and allows the services to respond to the events that comprise the FSM in a disciplined fashion.

The service logic components themselves are designed independently, and may be loosely coupled to other services by plugging into the service mediation components. If two services are highly dependent on each other, and cannot behave in a sequential fashion by plugging into generic service mediation mechanisms, then from the point of view of service mediation, they truly are one service. These highly dependent services should be represented by a single service component, which has the application-dependent code to decide how the two services will behave, given specific circumstances. The cause of these circumstances may be having a large number of variables that must be examined and compared, in order to determine what course of action to take. These variables may also be able to take on a large number of values. If each unique combination of values across the variables were to be treated by the service mediation as a discrete event, then the number of events could grow to be very large.

Time-of-day routing is an example of a service that may decide to take different actions based on precise values of date and time. If each unique combination of year, month, day, day of the week, hour of the day, minutes in the hour was considered to be a separate event handled by the service mediation of the service session, then the complexity of registering for events would be greatly increased. Services that did not care about these variables would be unduly burdened to take them into account, in order to register for large numbers of events based upon values that the service itself does not distinguish as separate events. A time-of-day service component could therefore plug into the service mediation, and would invoke time-sensitive service logic according to its application-dependent discriminator (namely, the time).

Therefore the selection of a practical set of mediated events is critical to developing a service mediation capability that will ease the burden of integrating third-party applications, instead of increasing the burden. These events should reflect the semantics of the FSM of the service session, which in turn may often reflect the superset of underlying call models and generic user actions. SAMM supports service mediation for call control events and activity events. Activity events do not come from call control, and they may or may not result in call control requests. The set of mediated events is a finite, published set of APIs that can be extended as needed.

It should also be noted that not all messages sent between objects need to be mediated events. Many notifications and transactions may occur between objects in the normal course of providing services. Only events that exhibit the following characteristics should be considered for inclusion as mediated events:

- Precedence must be established in processing the event in a sequential order (synchronous processing semantics) among at least some of the applications.
- An exclusive decision on what to do with the event must be determined according to the semantics of the service session FSM.

Service Mediation for SIP Service APIs

Services could be provided solely through SIP service APIs, SIP CPL, SIP CGI, and SIP servlets. The SIP services could be invoked by a SIP proxy server. When it receives a request for a particular address (subscriber), it could run the appropriate script. The difficulty with providing services solely via CPL is the limitations in combining a set of services, each created with a CPL script, with its own flowchart-style

logic. Combinations of the conditions expressed in nested flowcharts may become difficult to manage. A SIP servlet could invoke CPL scripts according to a precedence order specified using SAMM mediation. The precedence ordering could be provisioned, or could be determined according to a set of rules. The rules-based mechanism could require extensions to SIP.

SIP and Parlay/OSA

Parlay/OSA provides a network independent model. Parlay/OSA application servers may operate across wireless and wire-line networks. Parlay/OSA allows the customers to control network requests, and to request third-party call control. SIP and Parlay/OSA may be used to play complementary roles in providing services, with Parlay/OSA providing applications and call control, and SIP providing call/session signaling. Parlay/OSA provides a level of security, authorization, and authentication. The Parlay/OSA gateway may receive service requests through the Parlay/OSA API, and may signal the end users to set up communications, through the SIP proxy server.

Distributed Services and Service Mediation Example

1. A SIP INVITE is sent from user A toward user B through an outbound SIP proxy server.
2. The SIP proxy server runs a CPL script that gives possible endpoints for user B based on time of day.
3. The SIP INVITE routes through the network toward the first endpoint indicated for user B, a SIP softphone.
4. The inbound SIP proxy server that serves the SIP softphone runs a servlet that checks user B's presence through a presence server. The presence server indicates that user B is present on a PDA at this time, not on the SIP softphone.
5. The inbound SIP proxy server's servlet for user B forwards the SIP INVITE to an application server, which spawns a service session for user B. Refer to **Figure 10**.
6. The SIP INVITE message is translated into a transport neutral event of "incoming call," and then it is distributed by SAMM service mediation to a set of personalized call management services for which user B has registered.
7. The first service is a client stub for a privacy manager, which ensures that the calling party is properly identified. The client may send a message to the service, which runs on an SCP, across a Signaling System 7 (SS7) network. Since the calling party is properly identified, the privacy manager stub returns a value of "continue" to SAMM service mediation.
8. It passes the "incoming call" event to a screen list client stub, which runs a SIP CPL call screening script. The calling screening service determines that the calling party is allowed to alert user B, so the client stub returns a value of "continue" to SAMM service mediation.
9. It passes the "incoming call" event to the Parlay/OSA call control stub. The stub converts the event into a Parlay/OSA specific event, and notifies the appropriate Parlay/OSA application for user B. The Parlay/OSA application stub returns a value of "determined response," with a directive to alert user B.
10. A post-processing service, "custom alerting," implemented in JAIN, receives the event, and selects an image of user A from a database, to send to user B's PDA.

User B receives the image of user A along with alerting, and accepts the call.

Service Management

SAMM service management provides OAM&P capabilities for third-party programmable services, including life cycle management, measurements, billing, provisioning, and maintenance. Life cycle management deals with installing and activating new services, taking the services out of service, and de-installing them when they are no longer wanted. Refer to JAIN SLEE [25] for more discussions on service life-cycle management. SAMM is designed to allow plug in/plug out of services without impact on other services. SAMM also allows services to work with back office systems, to provision service data, to take measurements and generate billing, and to be managed by the maintenance system. For

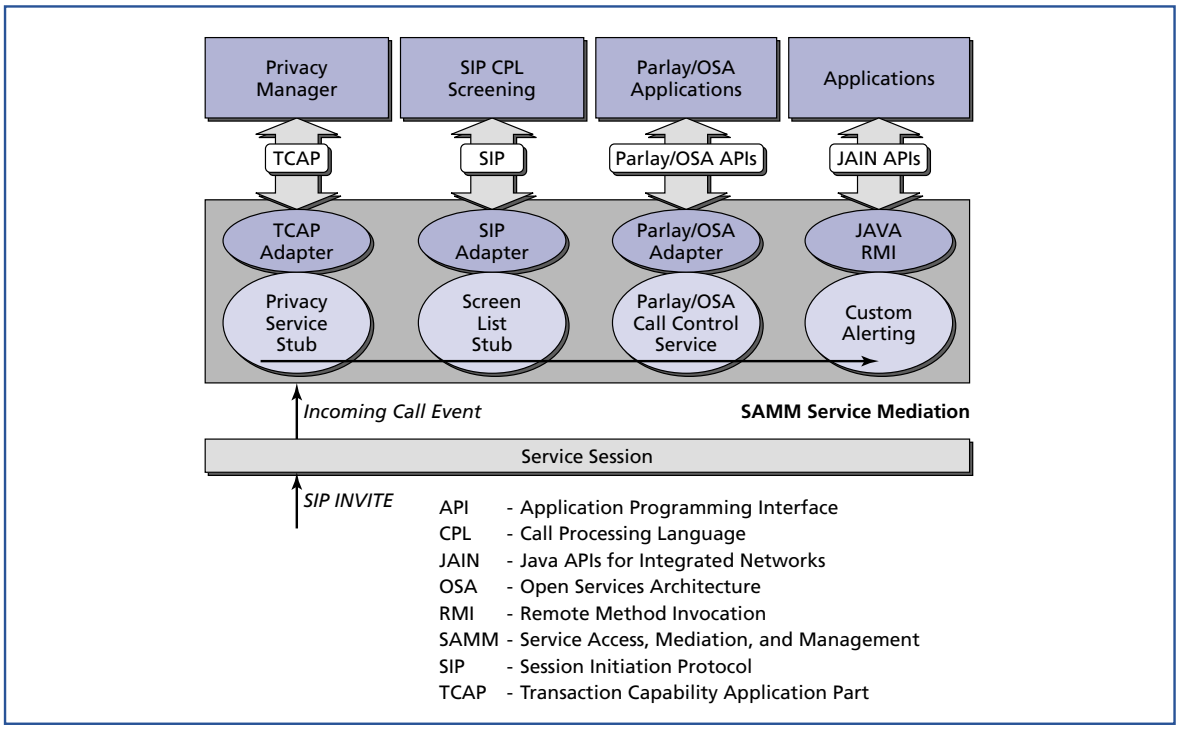

Figure 10.
Service mediation across multiple APIs.

more details on SAMM service management, refer to [2] and [3].

Conclusions

The development of a complete network-level services architecture requires a complete understanding of the types of services that are to be delivered, the best location in the network to deploy the services, and the network elements that are required to perform the services.

Acknowledgments

This paper is the product of the hard work and dedication of many people. We want to specifically acknowledge Bin-wen Ho and Troy Echols.

*Trademarks

CORBA is a trademark of Object Management Group, Inc.

JAIN and Java are trademarks of Sun Microsystems Inc.

Microsoft and Windows XP are registered trademarks of Microsoft Corporation.

References

[1] M. Day, J. Rosenberg, and H. Sugano, "A Model for Presence and Instant Messaging," IETF RFC 2778, Internet Engineering Task Force, Feb. 2000, <http://www.ietf.org/rfc/rfc2778.txt>.

[2] J. R. Dianda, R. O. Colbert, P. J. L. Herve, and T. Yang, "Programmable Service Platforms for Converged Voice/Data Services," Bell Labs Tech. J., 5:3 (2000), 43–58.

[3] J. R. Dianda, S. C. Darity, B.-W. Ho, and K. J. Scott, "Service Authoring for Third Party Programmable, Service Mediation Enabled Feature Servers in the Multiservice Core," Bell Labs Tech. J., 6:1 (2001), 192–210.

[4] I. Faynberg, L. R. Gabuzda, M. P. Kaplan, and N. J. Shah, The Intelligent Network

Standards—Their Application to Services, McGraw Hill, New York, 1997.
[5] I. Faynberg (ed.), J. Gato, H. Lu, and L. Slutsman, "SPIRITS Protocol Requirements," IETF Internet-Draft, Work in Progress, <http://www.ietf.org/internet-drafts/draft-ietf-spirits-reqs-04.txt>.
[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol—HTTP/1.1," RFC 2616, June 1999.
[7] N. Freed and N. Borensten, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," RFC 2045, Nov. 1996.
[8] V. Gurbani, "SIP-enabled IN Services: An Implementation Report," IETF Internet-Draft, Work in Progress, expired May 2001, <http://www.bell-labs.com/mailing-lists/ietf-sin/draft-gurbani-iptel-sip-in-imp-01.txt>.
[9] V. Gurbani and V. Rastogi, "Accessing IN Services from SIP Networks," IETF Internet-Draft, Work in Progress, expired Feb. 2002, http://www.bell-labs.com/mailing-lists/ietf-sin/draft-gurbani-iptel-sip-to-in-05.txt.
[10] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol," RFC 2543, Mar. 1999.
[11] Internet Engineering Task Force (IETF) Service in the PSTN/IN Requesting InTernet Service (SPIRITS) Working Group, <http://www.ietf.org/html.charters/spirits-charter.html>.
[12] Internet Engineering Task Force (IETF) SIP for Instant Messaging and Presence Leveraging (SIMPLE) Working Group, <http://www.ietf.org/html.charters/simple-charter.html>.
[13] Internet Engineering Task Force (IETF) SIP Working Group, <http://www.ietf.org/html.charters/sip-charter.html>.
[14] J. Kozik, I. Faynberg, and H. Lu, "On Opening PSTN to Enhanced Voice/Data Services—The PINT Protocol and Solution," Bell Labs Tech. J., 5:3 (2000), 153–165.
[15] J. Kozik, W. A. Montgomery, and J. J. Stanaway, Jr., "Voice Services in the Next-Generation Networks: The Evolution of the Intelligent Network and Its Role in Generating New Revenue Opportunities," Bell Labs Tech. J., 3:4 (1998), 124–143.
[16] J. Lennox and H. Schulzrinne, "Call Processing Language Framework and Requirements," RFC 2842, May 2000.
[17] H. Lu (ed.), I. Faynberg, J. Voelker, M. Weissman, W. Zhang, S. Rhim, J. Hwang, S. Ago, S. Moeenuddin, S. Hadvani, S. Nyckelgard, J. Yoakum, and L. Robart, "Pre-SPIRITS Implementation of PSTN-Initiated Services," IETF RFC 2995, Internet Engineering Task Force, Nov. 2000, <http://www.ietf.org/rfc/rfc2995.txt>.
[18] The PARLAY Group, Specifications, <http://www.parlay.org>.
[19] S. Petrack and L. Conroy, "The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services," IETF RFC 2848, Internet Engineering Task Force, June 2000, <http://www.ietf.org/rfc/rfc2848.txt>.
[20] J. Rosenberg, D. Willis, R. Sparks, B. Campbell, H. Schulzrinne, J. Lennox, C. Huitema, B. Aboba, D. Gurle, and D. Oran, "SIP Extensions for Instant Messaging," IETF Internet-Draft, Work in Progress, <http://www.ietf.org/internet-drafts/draft-ietf-simple-im-01.txt>.
[21] J. Rosenberg, D. Willis, R. Sparks, B. Campbell, H. Schulzrinne, J. Lennox, C. Huitema, B. Aboba, D. Gurle, and D. Oran, "SIP Extensions for Presence," IETF Internet-Draft, Work in Progress, <http://www.ietf.org/ internet-drafts/draft-ietf-simple-presence-04.txt>.
[22] H. Schulzrinne, L. Slutsman, and I. Faynberg, "Interworking between SIP and INAP," IETF Internet-Draft, Work in Progress, <http://www.bell-labs.com/mailing-lists/ietf-sin/draft-schulzrinne-sin-01.txt>.
[23] Session Initiation Proposal Investigation (SIPPING), IETF Working Group, <http://www.ietf.org/html. charters/sipping-charter.html>.
[24] L. Slutsman (ed.), I. Faynberg, H.-L. Lu, and M. Weissman, "The SPIRITS Architecture," IETF RFC 3136, Internet Engineering Task Force, June 2001, <http://www.ietf.org/rfc/rfc3136.txt?number=3136>.
[25] Sun Microsystems, Inc., "Java™ APIs for Integrated Networks," <http://java.sun.com/products/jain/index.html>.
[26] Sun Microsystems, Inc., "Java™ Servlet 2.3 Specification," Sept. 2001.
[27] Third Generation Partnership Project (3GPP), <http://www.3gpp.org/>.

(Manuscript approved April 2002)

JANET R. DIANDA is a distinguished member of

technical staff in Lucent Field Solutions in Overland Park, Kansas. She provides technical consultation in the areas of SIP and services architecture to the Sprint customer team. She authored and presented a paper, "A Multimedia Services Architecture for Converged Voice and Data Services," to the World Telecommunications Conference/International Switching Consortium. She has filed a United States patent for "Apparatus and Method to Manage the Invocation of Feature Service." Ms. Dianda holds an M.S. degree in computer science from the Illinois Institute of Technology in Chicago.

VIJAY K. GURBANI is a distinguished member of

technical staff in the Architecture and Next-Generation Evolution Department in Mobility Solutions at Lucent Technologies in Naperville, Illinois. He has been involved in Lucent's Internet Call Waiting offer and also in creating services on the Lucent Softswitch, mapping the IN Call Model to SIP, using SIP as an enabler for IN services, and implementing a SIP proxy server for demonstrating converged services. He has a B.Sc. and an M.Sc. in computer science from Bradley University, Peoria, Illinois, and is currently completing his Ph.D. research in the same field at the Illinois Institute of Technology, Chicago. His thesis is entitled "Enabling Services Through Protocol Interworking."

MARK H. JONES is a distinguished member of technical

staff in the Lucent Softswitch Architecture Department of Lucent's OPENet organization in Naperville, Illinois, where he is currently responsible for developing architectural visions for Lucent's next-generation converged voice/data networks. Dr. Jones holds a B.S. in electrical engineering from Michigan State University in East Lansing and M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign. ◆