

JAIN Protocol APIs

Ravi Raj Bhat and Rajeev Gupta, Trillium Digital Systems, Inc.

ABSTRACT

JAIN envisions the creation of a number of Java APIs that abstract the details of networks and protocol implementations, and allow for the development of portable applications. The JAIN Protocol Experts Group will focus on developing Java APIs for protocols used in telephony, INs, wireless networks, and the Internet. PEG is organized into an SS7 subgroup and an IP subgroup. The first section provides an introduction to PEG. The next section describes the JAIN SS7 APIs. We then describe the JAIN IP APIs. The article explains how JAIN SS7 and IP APIs can be leveraged for the converged SS7-IP networks of the future. We then describe the JAIN PEG roadmap and provide references.

AN INTRODUCTION TO PEG

Java APIs for Integrated Networks (JAIN) envisions the creation of a number of Java application programming interfaces (APIs) that abstract the details of networks and protocol implementations, and allow for the development of portable applications. The JAIN Protocol Experts Group (PEG) will focus on developing Java APIs for protocols used in telephony, intelligent networks (INs), wireless networks, and the Internet. PEG is organized into a Signaling System No. 7 (SS7) subgroup and an Internet Protocol (IP) subgroup. The SS7 subgroup will focus on developing Java APIs for SS7 technology, which is used in telephony, IN, and wireless networks. The IP subgroup will focus on developing Java APIs for Internet technologies. Within the SS7 and IP subgroups, there are Edit Groups that focus on specific protocols, such as the Transaction Capabilities Application Part (TCAP) and H.323 protocols. It is expected that the PEG Edit Groups will create several protocol APIs for public review in the year 2000. In addition, there will be associated reference implementations (RIs), conformance test suites (CTSs), demonstrable applications, and commercially available products.

JAIN SS7 APIS

SS7 technology provides a general-purpose, common-channel signaling system for use between network nodes in a telephony or wireless network. SS7 is optimized for operation in digital networks utilizing stored program-controlled exchanges. SS7 provides a reliable means of transferring information in correct sequence between network nodes without loss or duplication.

SS7 TELEPHONY

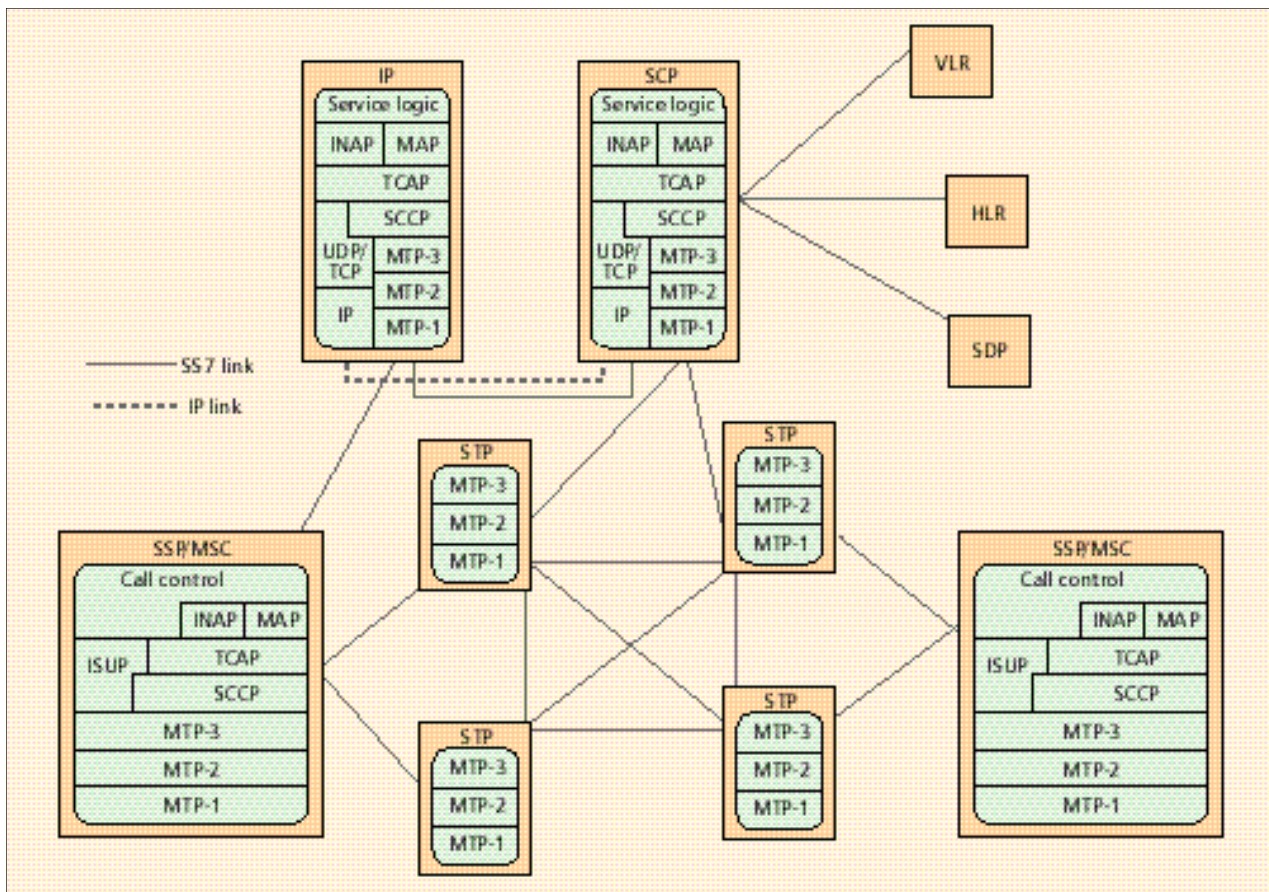
At the core of an SS7 network, the integrated services digital network (ISDN) User Part (ISUP) provides call signaling capabilities, and TCAP provides remote operation invocation capabilities via transaction exchange. Applications (e.g., call control, service logic) use the services provided by ISUP and TCAP to access the signaling capabilities of the SS7 network. ISUP and TCAP use the message delivery, routing, congestion control, link management, address translation, and application demultiplexing services provided by the Message Transfer Part (MTP) and Signaling Connection Control Part (SCCP).

INTELLIGENT NETWORKS

The call control application in basic SS7 networks has the service logic tightly coupled to the call processing logic in all switches, in the form of software. As a result, the software in all of the switches must be upgraded in order to provide new or improved services. This is a very expensive and time-consuming process, considering the number of switches in an SS7 network. The solution to this problem is to separate service logic from basic call processing and to concentrate the service logic in some dedicated entities in the SS7 network. Thus, when a new service has to be incorporated, or existing service logic has to be upgraded, only a few entities in the telecommunications network need be changed. In other words, the solution to the problem is to push the intelligence of the network to a few selected entities in the network. This effort enhances and reincarnates the basic SS7 network as an IN. IN classifies the SPs in an SS7 network into five different logical components:

- Service switching point (SSP), which provides switching, signaling, routing, and service invocation capability
- Signaling transfer point (STP), which provides signaling and routing capability
- Service control point (SCP), which provides service logic execution capability
- Service data point (SDP), which provides subscriber data storage and access capability
- Intelligent peripheral (IP), which provides resources such as customized voice announcements, voice recognition, and dual-tone multifrequency (DTMF) digit collection

SSP, SCP, and IP communicate using the IN Application Part (INAP) protocol layer to invoke and execute service logic.



■ Figure 1. The integrated SS7, IN, and wireless network framework.

WIRELESS NETWORKS

Existing and emerging wireless networks are built on top of the basic framework provided by the IN, by introducing the new Mobile Application Part (MAP) protocol layer over TCAP. For example, Global System for Mobile Communications (GSM) defines GSM-MAP over TCAP. In addition, wireless networks introduce both the home location register (HLR, permanent storage for subscriber data) and the visited location register (VLR, temporary storage for subscriber data). These two entities provide the same service as SDP. The SSP (renamed mobile switching center, MSC, in GSM), SCP, VLR, and HLR communicate using MAP to invoke and execute mobility-specific service logic. Figure 1 illustrates various protocols involved in an integrated SS7, IN, and wireless network framework.

INTELLIGENT NETWORK SERVICES

At the service level, the IN can be described in terms of various logical components:

- Service-independent building block (SIB), a standard reusable networkwide capability used to realize services and service features
- Basic call process (BCP), a standard SIB, which identifies the normal call process and invokes IN services
- Global service logic (GSL), which describes how SIBs are chained together to provide service features

- Point of initiation (POI), which is the point in the BCP when control is passed over from the BCP to the chain of SIBs implementing a service
- Point of return (POR), which is a point in the BCP when control is passed back from the chain of SIBs after the service is completed

A sample credit card calling (CCC) service execution is illustrated in Fig. 2. This figure also explains various steps involved in providing CCC service.

THE JAIN SS7 API: IN SERVICE FACILITATOR

Substantial investment has been made in writing software for the SS7 protocol layers in the current network infrastructure. Most of this software is written in native programming languages, such as C, which hinders fast and easy introduction of new services in INs and wireless networks. This is because programs written in native languages cannot be easily uploaded to a running system. To upgrade the service logic in an SCP, usually one must shut down the SCP, install the new software, and then bring it back online. Java, as an object-oriented programming language (inherently supporting distributed, platform-independent computing), provides the framework to easily upload new objects (implementing various service components) into the existing infrastructure. In order to utilize Java technology for this purpose, new service applications written in Java must be adapted to commu-

nicate with protocol layers that are implemented in native programming languages, such as C. The JAIN SS7 API is an effort toward providing such an adaptation tool.

Due to various market forces and regional requirements, present telecommunications networks embody a myriad of variants for each protocol layer. Whenever a new protocol variant is introduced into the network to provide a new bearer service or some other advanced network-level service, application-level software entities must be upgraded to work with new protocol variants. The JAIN SS7 API is an attempt to abstract the network-level functionality from the idiosyncrasies of different protocol variants so that changes in the underlying networks are transparent to the application. Figure 5 illustrates how various components defined by JAIN will fit into the current network to create a flexible future network.

At its core, the JAIN SS7 API architecture defines a set of software components that enable an application (e.g., service execution environment in an SCP and call control in an SSP) executing in the Java space to access services provided by underlying SS7 protocol layers written in a native programming language. IN service components, such as SIBs, are analogous to objects, or JavaBeans [1]. The service creation and management center shown in Fig. 5 will cre-

ate and upgrade the SIBs (illustrated in Fig. 2) in Java and upload them at the service execution environment in real time using either the Java Dynamic Management Kit (JDMK) or Enterprise JavaBeans environment. In addition, the JAIN SS7 API provides an operation, administration, and maintenance (OA&M) API to administer and manage various SS7 protocol layers.

The software components defined in the JAIN SS7 API are based on the JavaBeans design pattern. Each JAIN SS7 API defines three major Java software components, as shown in Fig. 3: Stack, Provider, and Listener. These three software components are defined in the form of Java interfaces. The Stack component abstracts the underlying native SS7 protocol stack, and provides a "factory" to create and manage the Provider component, which exchanges protocol messages with the native SS7 protocol layer using a proprietary interprocess communication (IPC) mechanism. Therefore, Provider components are specific to a particular implementation of a protocol layer. Listener components interact with the Provider component via Event objects. To exchange Event objects, Listener components must register with Provider components. Listener components should be portable across various implementations of Provider components. Applications use

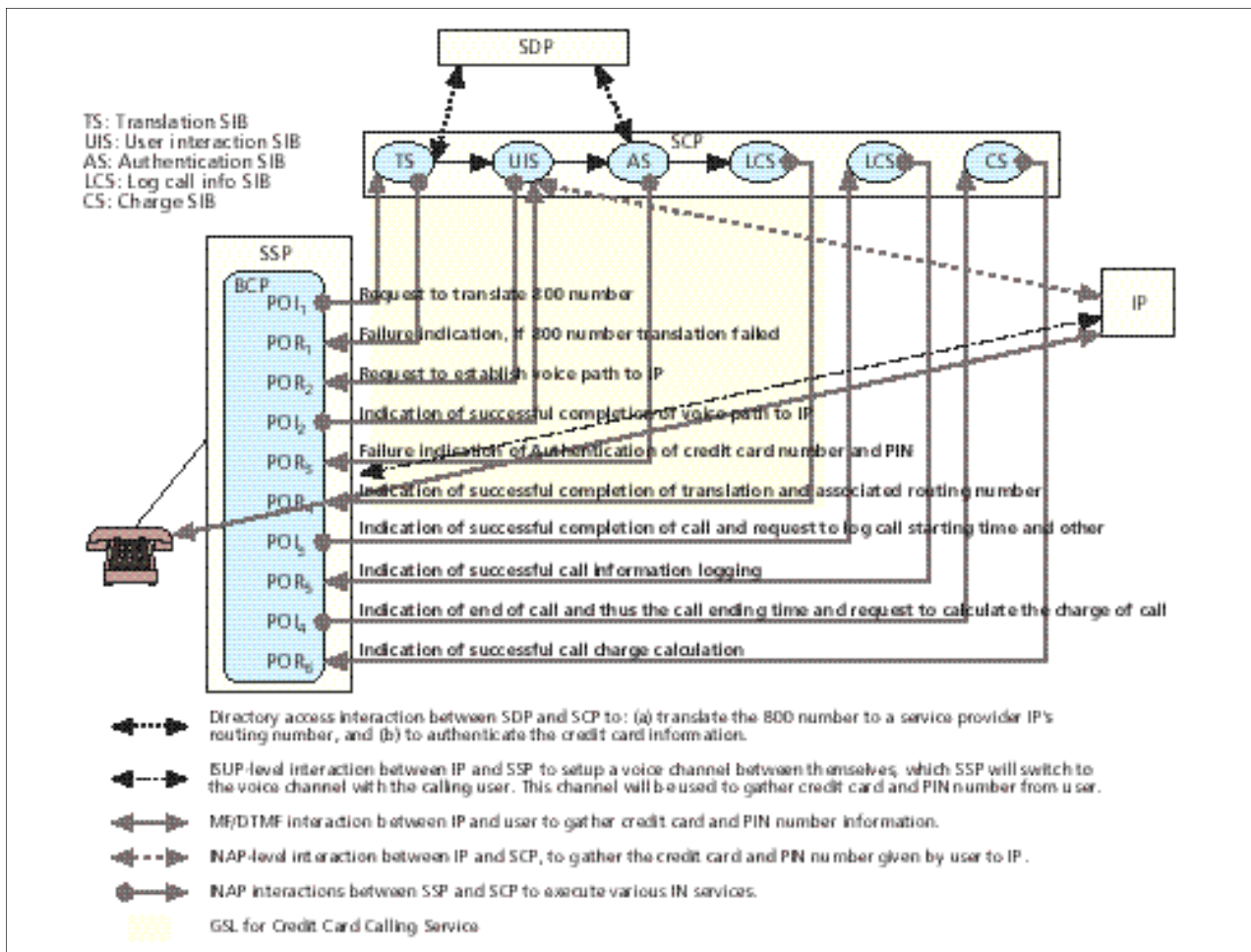
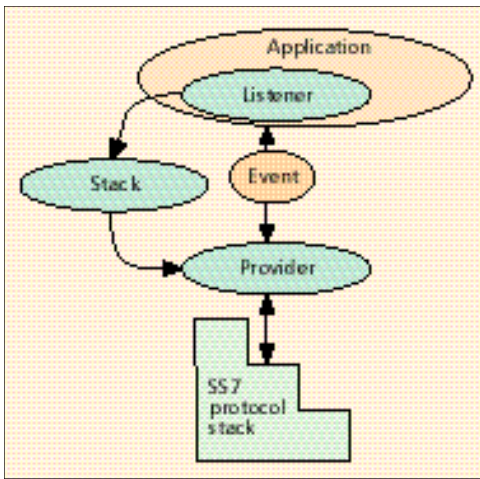


Figure 2. BCP and SIB interaction to provide credit card calling service.



■ Figure 3. The JAIN SS7 API architecture.

the JAIN SS7 API to implement a Listener interface. The Provider component maps the generic API to a specific flavor of a protocol, implemented by the native protocol stack.

The JAIN SS7 API defines Java-based APIs for the following protocols:

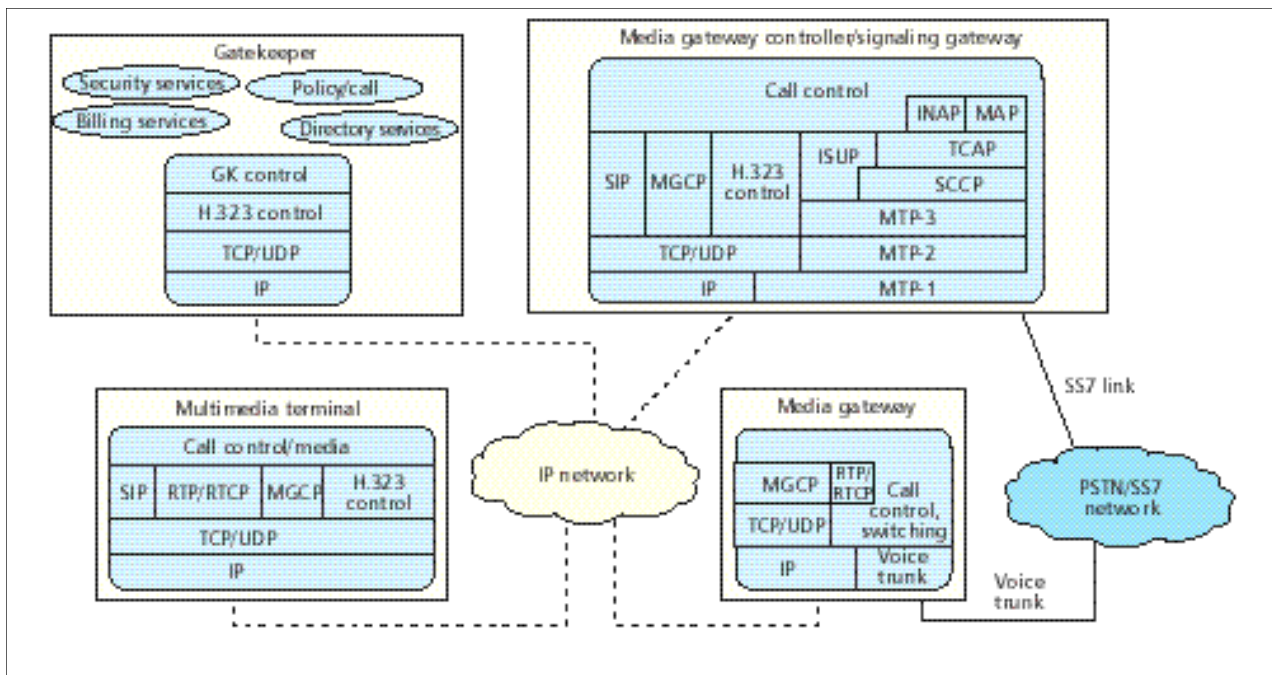
- TCAP – a non-call-related transaction-based control protocol that provides support for message exchange between interacting applications in a distributed environment, such as the SS7 network infrastructure. The applications at signaling points (SPs) exchange messages using transactions and components. Transactions represent a dialog (or a series of message exchanges) between the applications. Components represent a specific operation requested by an application or the result of an operation requested by an application (in the context of a dialog). The JAIN TCAP API specifies the Java interfaces and classes that are required for initializing and terminating a TCAP transaction, managing TCAP dialog identifiers, and building and sending TCAP transactions and components. Using the JAIN TCAP API, applications and the native TCAP protocol stack can exchange TCAP transactions and components in the form of Java Event objects. The JAIN TCAP API will be based on the ITU'93 [2], ITU'97 [3], ANSI'92 [4], and ANSI'96[5] variants of the TCAP protocol. The JAIN TCAP API is currently available for public review. It was publicly released at the end of the fourth quarter of 1999.
- ISUP – a call-related signaling protocol that provides support for call establishment and release and trunk circuit management in an SS7 network. Using the JAIN ISUP API, call control applications can exchange ISUP control messages with native ISUP protocol stacks in the form of Java Event objects. The JAIN ISUP API will be based on the ANSI'92, ANSI'96, ITU'93, and ITU'97 ISUP specifications. The JAIN ISUP API is currently in the requirements study phase and is expected to be available for public review in the first quarter of 2000.

- MAP – a non-call-related control protocol that provides support for interacting mobile applications in a distributed network environment. The JAIN MAP API will be based on the European Telecommunications Standards Institute (ETSI) MAP GSM and Telecommunications/Electronics Industry Associations (TIA/EIA) IS41 specifications. Using the JAIN MAP API, mobile applications can establish and release basic dialog and perform remote operation invocation between two network entities. The JAIN MAP API is currently in the requirements study phase and is expected to be available for public review in the first quarter of 2000.
- INAP – a non-call-related control protocol that allows applications to communicate between various nodes/functional entities of an IN. The protocol defines the operations required to be performed between SPs for providing IN services, such as number translation, time of day, and follow me. The JAIN INAP API will be based on the American National Standards Institute (ANSI)/Bellcore Advanced Intelligent Network (AIN 0.2) and International Telecommunication Union — Telecommunication Standardization Sector (ITU-T) CS-2 INAP specifications. The JAIN INAP API is currently in the requirements study phase and is expected to be available for public review in the first quarter of 2000.
- OA&M – a standard mechanism used to provision and maintain various protocol entities in the SS7 network. OA&M includes the configuration of options for hardware, routing, transmission rates, and circuits. While other JAIN SS7 APIs define a generic interface in order to exchange protocol messages between applications and native implementations of protocol stacks, the JAIN OA&M API defines a generic interface to manage various components in the native implementations of protocol stacks. The JAIN OA&M API defines a managed object (MO) for each manageable component in the SS7 network. MOs could be created and managed using tools such as JDMK, which is based on Java Management Extension (JMX). Using the JAIN OA&M API, network management entities can easily provision and monitor various components in native implementations of protocol stacks. The JAIN OA&M API is currently available for public review. It was publicly released at the end of the fourth quarter of 1999.

A Brief Look at JAIN TCAP API — The JAIN TCAP API[5] is defined as a Java package, `jain.protocol.ss7.tcap`. Within this package, the JAIN TCAP API specification describes the following Java components:

- `jain.protocol.ss7.tcap.JainTcapStack`: A Java interface, to be implemented by TCAP protocol stack vendors, to represent their implementation of the TCAP stack for provisioning and management
- `jain.protocol.ss7.tcap.JainTcapProvider`: A Java interface, to be implemented by TCAP protocol stack vendors, to interact with their implementation of the TCAP stack

The JAIN SS7 API is an attempt to abstract the network-level functionality from the idiosyncrasies of different protocol variants so that changes in the underlying networks are transparent to the application.



■ Figure 4. An IP network supporting multimedia traffic.

- `jain.protocol.ss7.tcap.JainTcapListener`: A Java interface, to be implemented by TCAP application vendors (typically for IN and wireless services), to access TCAP services via any implementation of a `JainTcapProvider` interface
- `jain.protocol.ss7.tcap.Component`: A Java package, with a standardized set of classes and associated methods, to build various TCAP components
- `jain.protocol.ss7.tcap.Dialog`: A Java package, with a standardized set of classes and associated methods, to build various TCAP transactions

A factory design pattern is one in which a specific class is defined to create object instances of other classes whose implementation is hidden. As a part of the JAIN SS7 API, a factory class, `jain.protocol.ss7.JainSS7Factory`, is defined and used to create an object instance of the class implementing `JainTcapStack`. Protocol stack vendors will provide a class `com.<CompanyName>.jain.protocol.ss7.tcap.JainTcapStackImpl`, which implements `JainTcapStack` interface, and a class `com.<companyName>.jain.protocol.ss7.tcap.JainTcapProviderImpl`, which implements `JainTcapProvider` interface.

The following is the sequence of events that occurs after the TCAP application (e.g., call control) is instantiated and before the first component is sent to the native TCAP stack:

- Using the `JainSS7Factory.getInstance()` static method, the TCAP application creates an object instance of `JainSS7Factory` class.
- Using the `JainSS7Factory.setPathName()` method, the TCAP application sets the path where JAIN TCAP API classes are defined to `com.<CompanyName>.`
- Using the `JainSS7Factory.createSS7Object()` method, the TCAP application creates an

object instance of `com.<CompanyName>.jain.protocol.ss7.tcap.JainTcapStackImpl` class.

- Using the `JainTcapStackImpl.createAttachedProvider()` method, the TCAP application creates an object instance of `com.<CompanyName>.jain.protocol.ss7.tcap.JainTcapProviderImpl` class. During the creation of this object, it attaches itself to the native protocol stack.
- Using the `JainTcapProviderImpl.addJainTcapListener()` method, the TCAP application registers with the `JainTcapProviderImpl` class.
- Using the `JainTcapProviderImpl.getDialogId()` method, the TCAP application obtains a new dialog ID to initiate a new dialog.
- Using the constructors of an appropriate subclass of `ComponentReqEvent`, the TCAP application creates an object instance of a TCAP Component and populates its attributes. It then sends this Component toward the native TCAP Stack using the `JainTcapProviderImpl.sendComponentReqEvent()` method.
- Using the constructors of the appropriate subclass of `DialogReqEvent`, the TCAP application creates an object instance of a TCAP Dialog and populates its attributes. It then sends this Dialog toward the native TCAP stack using the `JainTcapProviderImpl.sendDialogReqEvent()` method. This completes the initiation and transmission of a dialog.

JAIN IP APIS

The ease with which data travels over the Internet at low cost has inspired industry pioneers to carry traditional voice and multimedia traffic over the Internet. To provide services over an IP network equivalent to those of a traditional telephone network, various new protocols, such as

H.323, Media Gateway Control Protocol (MGCP), Session Initiation Protocol (SIP), and various new network entities, such as media gateways, gatekeepers, call agents, and media gateway controllers, have been defined. The operation of these entities is depicted in Fig. 4.

This picture depicts various types of multimedia communications over an IP network, using different protocols. In a real deployment scenario, only a subset of these protocols and devices will be used:

- **Multimedia terminal:** The multimedia terminal is directly connected to the IP network, and contains an implementation of the multimedia control and media protocols. The terminal could be a PC with special software, or a dedicated appliance, such as an IP phone. An H.323 terminal will contain an implementation of the H.323 Control protocol for setting up multimedia calls. Alternative protocols for setting up multimedia calls are SIP and MGCP. In all cases, Real-Time Transport Protocol (RTP) and Real-Time Transport Control Protocol (RTCP) are used for managing media flows.
- **Media gateway:** In order to connect the IP telephony network with the existing PSTN, there is a need for a gateway between the two networks. A media gateway (MG) connects the public switched telephone network (PSTN) voice trunks to the IP network. It is a relatively simple device that translates circuit-switched voice to packet voice. In addition to terminating the voice circuit, it contains RTP/RTCP for sending packet data into the IP network and MGCP to communicate with an intelligent controller in the IP network.
- **Media gateway controller/call agent:** The MG controller (MGC) is complementary to the MG. It provides control connectivity to the PSTN via an SS7 link, terminating SS7 ISUP call signaling. It controls the MG through MGCP to set up connections between the PSTN and the IP network. Furthermore, it communicates with other call signaling elements in the IP network using protocols such as H.323 and SIP. This element represents the call signaling intelligence in the IP network. In order to move this intelligence away from the edge of the IP network, it is possible to split this gateway into a signaling gateway that terminates SS7 signaling and then transports it to an intelligent controller, the call agent (CA), located inside the IP network, using a protocol such as Signaling Common Transport Protocol (SCTP).
- **Gatekeeper:** The gatekeeper (GK) is an H.323 Control entity that controls access to resources, implements policies, and provides access to other "back-end" services, such as directory lookups for call routing. It may be implemented as a standalone entity or combined with the MGC. In the latter case, it could use the SS7 link to access IN services located inside the PSTN, such as access to number translation databases and subscriber information databases.

There are a wide variety of devices and platforms that can be used to provide multimedia

services over IP networks, depending on the combination of functions deployed in a single device. However, the services that are required by end users are well known; for example, for telephony, services such as call waiting, caller identification, 800-number translation, and calling cards are required. It is desirable to allow the services to be written in Java so that they can migrate across a variety of platforms as IP telephony networks and technologies evolve. By defining Java APIs to use native protocol software, the JAIN IP API subgroup seeks to preserve the existing native protocol software in these network entities, while simultaneously enabling new Java applications to provide new services. The JAIN IP API will define Java-based APIs for the following protocols:

- **H.323** – provides support for the transmission of real-time audio, video, and data communications over packet-based networks. H.323 specifies the components, protocols, and procedures providing multimedia communication over packet-based networks, such as IP or Internet Packet Exchange (IPX)-based local area networks (LANs), enterprise networks (ENs), metropolitan area networks (MANs), and wide area networks (WANs). H.323 defines call setup, the exchange of compressed audio and/or video, conferences, and interoperability with non-H.323 endpoints. The JAIN H.323 API will be a standard, generic Java-based interface to native H.323 protocol stacks. The JAIN H.323 API will facilitate the exchange of various H.323 control messages, such as H.225 call signaling, H.245 control signaling, and H.225 Registration, Administration, and Status (RAS) services. The JAIN H.323 API is currently in the requirements study phase and is expected to be available for public review in the second quarter of 2000.
- **MGCP** – defines the messages to be exchanged between the CA and MG. The CA uses these messages:
 - To instruct the MG to watch for specific events such as hook actions or DTMF tones on a specified endpoint
 - To create and release connections
 - To modify connections
 - To audit connections
 MGs use these messages to notify CAs whenever an event, such as a call, is received. Therefore, the JAIN MGCP API will be a standard, generic Java-based interface that sends and receives MGCP control messages to and from the underlying native protocol stacks in the MG, and to and from call control applications in the CA. The JAIN MGCP API will be based on the MGCP Internet drafts and the Cable Labs Packet-Cable Network-Based Call Signaling Protocol Specification. The JAIN MGCP API is currently in the requirements study phase and is expected to be available for public review in the first quarter of 2000.
- **SIP** – a text-based, transport-independent protocol that enables the transport of multimedia traffic over an IP network. SIP is

By defining Java APIs to use native protocol software, the JAIN IP API subgroup seeks to preserve the existing native protocol software in these network entities, while simultaneously enabling new Java applications to provide new services.

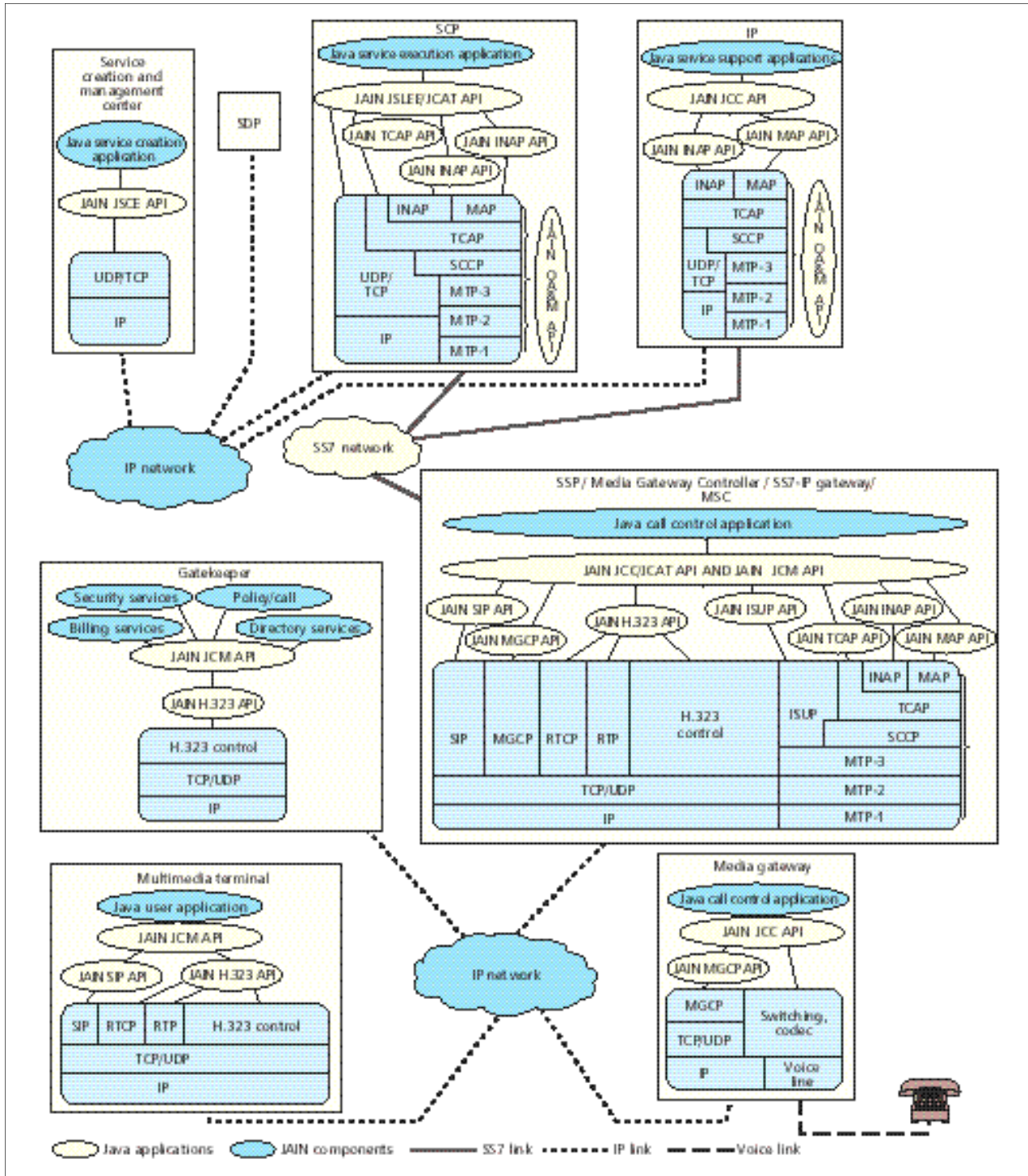
based on a client-server model, where SIP clients issue requests and the SIP server responds to the request.

The JAIN SIP API will be a standard, generic Java-based interface that sends and receives SIP control messages to and from the native SIP protocol stacks in SIP clients and SIP servers. The JAIN SIP API will be based on RFC 2543, published by the Internet Engineering Task Force (IETF). The JAIN SIP API is currently in the requirements

study phase and is expected to be available for public review in the second quarter of 2000.

SS7 AND IP CONVERGENCE

Communications networks are rapidly converging. There is tremendous interest in using packet-based networks for carrying all types of traffic including voice, data, video, and multimedia. At the same time, the legacy of traditional circuit-



■ Figure 5. An integrated Java-enabled multimedia network.

based networks remains. To provide truly seamless connections, it is necessary to interwork between circuit-based PSTN and the emerging voice-over-packet (VoP) networks. This interworking is achieved using elements, such as gateways and gatekeepers, that straddle network boundaries and allow media information, call signaling, and services to flow across the diverse networks.

JAIN is ideally suited to enable applications to migrate smoothly across diverse networks. As the JAIN APIs abstract the network details, the applications become network-agnostic. The following examples will illustrate the applicability of JAIN in this context:

- One example relates to access of IN services, such as 800-number translation, from voice-over-IP (VoIP) networks. The multimedia terminal (VoIP client) initiates a call to an 800-number. Existing telephony networks have a tremendous amount of data already available in their databases, or SCPs. The VoIP call request is typically routed by a gatekeeper element in the VoIP cloud, which will then need to access data in the SCPs. Such access is typically achieved using the SS7 TCAP protocol. However, a gatekeeper application written in Java could use a JAIN TCAP API to communicate with the database. The TCAP protocol can run over traditional SS7 transport, requiring the gatekeeper to have an SS7 interface. In Fig. 5, this would imply that the gatekeeper and the signaling gateway are collocated. Alternately, TCAP can run over an IP network, using SCTP, to a signaling gateway, which then provides access to the database. Thus, in this scenario, the gatekeeper application can provide SS7-based services while running on top of an IP network.
- Another example involves the use of an IP transport network for IN services. In this case, the traditional IN elements such as SCP, SDP, service creation environment (SCE), service management system (SMS), and IP are not connected over traditional SS7 transport; rather, they are connected to an IP network. TCAP runs over TCP/IP and serves to carry the IN messages over the IP network. The SCP and IP still have an SS7 interface in order to connect to the PSTN equipment. However, IN service creation, administration, and execution can be done over the IP network at the back end. For example, the SCP with the SS7 interface could have the JAIN TCAP API implemented. However, the service logic that uses the API can execute in a remote server using an abstraction such as Java Remote Method Invocation (RMI) or Common Object Request Broker Architecture (CORBA) that runs over the IP network. This allows the service modules to be distributed across the network, instead of inside a single device, and the modules can be upgraded dynamically without affecting the operation of other service modules.

As network technologies and protocols proliferate, the ability to write Java-based applications

that are portable across various networks will become increasingly attractive and cost-effective.

ROADMAP FOR JAIN PEG

PEG started with one Edit Group for SS7 TCAP API, and has expanded to include eight Edit Groups. As new network technologies emerge and new markets develop, it is conceivable that there will be greater demand for more PEG-standardized protocol APIs. The current and future focus for PEG remains the integration of circuit- and packet-based networks. Protocols for telephony, wireless, and the Internet will also remain of primary interest to the PEG.

In addition, once the initial standard APIs are released, there will continue to be ongoing efforts to keep them updated with the latest advances in technology. Protocols for packet technologies are rapidly developing, including protocols for transport (e.g., Signaling Transport, SIGTRAN), high-speed routing (e.g., multi-protocol label switching, MPLS), security, and quality of service. New capabilities are being developed for intelligent networks and third-generation wireless networks. Even the traditional SS7 call signaling technology presents new challenges in the form of country-specific variants. Finally, managing the various networks, protocols and services requires enhanced OA&M APIs. So, in conclusion, the PEG has a difficult, but exciting, road ahead.

REFERENCES

- [1] JavaBeans API Specification (v1.01), Sun Microsystems Inc., July 1997.
- [2] ITU-T Rec. Q.771-Q.775, "Transaction Capabilities Application Part (TCAP)," Mar. 1993.
- [3] ITU-T Rec. Q.771-Q.775, "Transaction Capabilities Application Part (TCAP)," June 1997.
- [4] "ANSI Signaling System Number 7 (SS7) — Transaction Capabilities Application Part (TCAP)," T1.114, Sept. 1992.
- [5] "ANSI Signaling System Number 7 (SS7) — Transaction Capabilities Application Part (TCAP)," T1.114, Mar. 1996.
- [6] "JAIN TCAP API Specification, Release 1.0, Public Review Draft," Sun Microsystems Inc., June 1999.

BIOGRAPHIES

RAVI RAJ BHAT (raviraj@trillium.com) is leading the SS7 Telephony and Java development group within Trillium as an Engineering Project Manager and is responsible for the research and development activities in the area of SS7 telephony and Java. He joined Trillium in May 1995 and since then has been involved in research and development activities in ATM, SS7, intelligent networks, 3G wireless systems, and virtual private networks. Prior to that he worked in the research and development department of Wipro Infotech Ltd. in India. He received a Bachelor's degree in computer engineering from the Karnataka Regional Engineering College, India, in 1993. He is currently pursuing an M.B.A. degree from the Anderson School of Management at the University of California, Los Angeles.

RAJEEV GUPTA (rajeev@trillium.com) has served as engineering director of the Emerging Technologies Department of the Engineering Group since May 1997 and is responsible for conducting research to keep Trillium at the forefront of communications technology. He has also served as strategic planning director with the Marketing and Sales Group since September 1998. He joined Trillium in January 1993 as a member of technical staff. From 1993 to May 1997 he held various positions in Trillium's Engineering Group. He received a Bachelor's degree in computer science and engineering from the Indian Institute of Technology (IIT), Delhi, in 1990. He earned a Master's degree in computer science from the University of California, Los Angeles, in 1992.

The current and future focus for PEG remains the integration of circuit- and packet-based networks. Protocols for telephony, wireless, and the Internet will also remain of primary interest to the PEG.