ELSEVIER

Computer Communications 23 (2000) 740–753

computer communications

www.elsevier.com/locate/comcom

Advanced service architecture for H.323 Internet Protocol Telephony

J. Tang[a], T. White[a], B. Pagurek[a,*], R. Glitho[b]

[a]*Systems and Computer Engineering, Carleton University, Ottawa, Canada*
[b]*Ericsson Research Canada, Montreal, Canada*

Abstract

Internet Protocol (IP) based communication is fast becoming a viable alternative for voice communications. The Intelligent Network (IN) represents the world wide accepted basis for the uniform provision of advanced telecom services. Mobile Agents offer unique opportunities for structuring and implementing open distributed service architectures, facilitated by the dynamic downloading and movement of service code to specific network nodes. In this paper, a new service architecture for IP telephony, based on the ITU-T standard H.323, is proposed. The implementation uses Mobile Agents as an enabling technology and existing architectural concepts taken from IN. This IP service architecture enables telecom services deployed through mobile service agents on a per user basis, which results in several advantages when compared to centralized service architectures.

Keywords: IP telephony; Mobile agents; H.323 supplementary services

1. Introduction

In February 1995, Internet Protocol Telephony [1–3] or IP telephony, which delivers voice and data [4] communications over IP networks, became a reality for the first time when Vocaltec, Inc. introduced its Internet Phone software [3]. Because of its low price and efficient use of bandwidth, it has progressed rapidly in the relatively short period of time since then. As the Internet is an open, distributed and evolving entity, it is expected that there will be many extensions to IP telephony. In the classical telephony world which is based on circuit switched networks, a number of service architectures have been developed in the last decade; for example, the Intelligent Networks (IN) framework, Telecommunications Management Network (TMN), Telecommunications Information Networking Architecture (TINA), etc. The purpose of these service architectures is to increase the quality and range of services offered in communication networks. In order to compete with classical telephony in today's market, one of the challenges that IP telephony faces is to offer not only the same high quality voice calls, but also a set of call features (i.e. advanced services) that classical telephony offers today. While the high quality of voice calls has yet not been achieved in the IP telephony world, sound architectures are needed for the control and management of services.

The traditional telephone system has very primitive end-terminals (telephones) and considerable intelligence inside the network [5]. Advanced service architectures separate call setup and call processing functions. In general, the Internet represents a different balance, with intelligent end-terminals (computers) and a simple set of functions inside the switches of the network. Switches are composed of software and general-purpose hardware. It is reasonable to foresee that long-term evolution of IP telephony will have much more intelligence implemented in the end terminals rather than inside the network. Advanced services such as call diversion and call transfer, which are implemented inside the telephone network today, can be implemented in users' computers.

In order to realize this view of IP telephony, appropriate protocols and technologies are needed. Currently, there are two protocols that address this issue, one is International Telecommunication Union–Telecommunication Standardization Sector (ITU-T) H.323 [6] and the other is the Internet Engineering Task Force (IETF) Session Initiation Protocol (SIP). The first version of H.323 specification was approved in 1996 by the ITU-T's Study Group 16. Version 2 was approved in January 1998. A third version of the standard is planned for approval in 1999. The standard is broad in scope and includes both stand-alone devices and embedded personal computer technologies as well as point-to-point and multi-point conferences. SIP is rather lightweight, reusing many of the header fields, encoding rules, error codes, and authentication mechanisms of HTTP (see, for example Refs. [7,8] for a comparison and Ref. [9] for a critique). While service architecture requirements for

* Corresponding author.
E-mail address: bernie@sce.carleton.ca (B. Pagurek).

0140-3664/00/$ - see front matter
PII: S0140-3664(99)00234-0

IP telephony have yet to reach maturity, a set of requirements for Telecommunications Information Networking Architecture Consortium (TINA-C) [10] service architecture described in Ref. [11] are gradually being adopted. In this report, we use them to evaluate the service architecture defined in H.323, and the new service architecture described later. The Service life cycle, defined by TINA-C service architecture recommendation, consists of service construction, deployment, utilization and withdrawal. Each stage of the service life cycle is explained in Ref. [11]. The requirements used to explore IP telephony advanced service architectures can also be found in Ref. [11].

Supplementary services supported by H.323 are specified in H.450.x. Each of the defined supplementary services has its own specification. As pointed out in Ref. [8], "How they may be broken down into reusable building blocks is not clear and this will lead to specification and implementation inefficiency." The H.323 specification does not address service control and management. In addition, there are no third party defined services. Hence there is obviously room left for developers to design more flexible service architectures using enabling technologies; e.g. Mobile Agents (MA).

IP telephony is real-time data communications over IP transport. As the Internet is an open, distributed and evolving entity, flexibility, scalability and robustness are very important issues to be considered when designing new service architectures. MA technology has the ability to provide solutions addressing all of these issues. As pointed out in Ref. [12], the key advantage of MA is flexibility. It can enhance service architectures, providing easy service customization and instant service provisioning.

The remainder of this paper is organized as following. In Section 2, the basic concepts of IN and MA, Jini/JavaBeans, and advantages for using them, are explained. In Section 3, a new IP telephony service architecture and aspects of the implemented solution using MA, Jini and IN concepts are described. Section 4 presents three scenarios for application of this new service architecture—Call Forwarding, Call Transfer and Virtual Private Network (VPN) Services. Section 5 presents the conclusions relating to this research.

2. Enabling technologies

The Java language, developed by Sun Microsystems, has a number of advantages that make it particularly appropriate for MA technology. Its main appeal for agents is its portability, the use of bytecodes and its interpreted execution environment. This means that any system with sufficient resources can host Java programs.

The Java Virtual Machine and Java's class loading model, coupled with several of Java features—most importantly serialization, remote method invocation, multithreading, and reflection—have made building first-pass MA systems a fairly simple task [13].

MA, as one of the enabling technology for IP telephony services, is introduced in Section 2.1. JavaBeans and Jini (both from Sun Microsystems) are two good complementary candidate technologies for the implementation of an IP telephony service architecture; they are described in Section 2.2.

2.1. Mobile agent technology

Software agents originated with Distributed Artificial Intelligence (DAI) [14] research, the concept of an agent can be traced back to the early days of 1970s. There is a broad range of companies and universities that are actively pursuing agent technology and the number is growing steadily. The use of intelligent agents in Telecommunications has a number of documented advantages [12,15].

An agent can be described as a software component that performs a specific task autonomously on behalf of a person or an organization [16]. It contains some level of intelligence, ranging from predefined rules to self-learning Artificial Intelligence (AI) mechanisms. Thus agents may operate rather asynchronously to the user and may communicate with the user, system resources and other agents as required to perform their tasks. They are often event or time triggered.

A mobile agent is one of the seven agent types identified in Ref. [14]. An agent is an object with its private thread of execution, also known as an "active object" [17]. An MA is the kind of agent that is not bound to the host where it begins execution. It has a unique ability to transport itself from one host in a network to another. As it travels, it performs work on behalf of a network user. Agent mobility is probably the most challenging agent property, in that it provides an intelligent agent with the potential to influence the traditional way of communications and service realization. Customizability is the result of the diffusion of network services and applications. It allows users to tailor services according to their specific needs and preferences. Flexibility and extensibility are due to the dynamic nature of the underlying network infrastructure and service demand [18]. Other arguments for mobile agents have also been forthcoming [17,19].

In the past, the main motivations for the application of mobile agents were the lack of capacity to execute programs locally, and the desire to share resources and improve load balancing in a distributed system. In contrast to these concepts designed for rather specific or closed environments, new agent concepts aim for open environments (e.g. the Internet). Today, flexibility is a key design issue for emerging network service architectures in order to adapt quickly to the changing customer service demands. The following are some of the reasons for using MA technologies:

- An MA-based approach may reduce the network load when compared to a Remote Procedure Call (RPC)-based approach.
- Asynchronous and autonomous execution provide the possibility for realization of advanced services by means of using mobile agents.

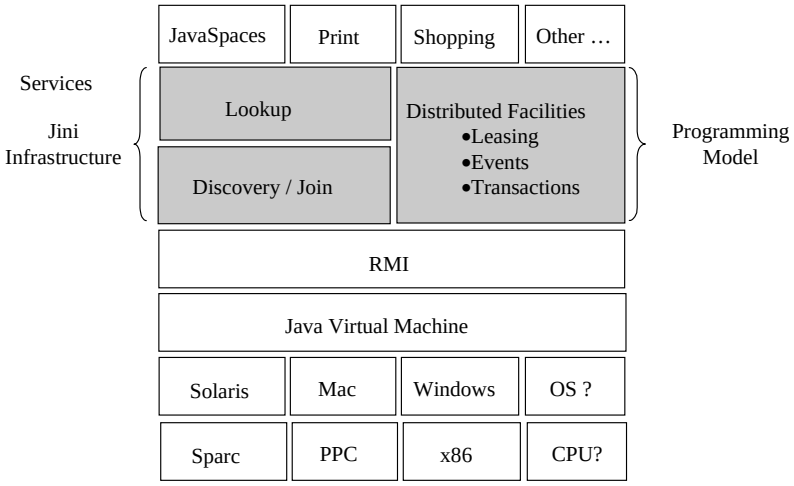


Fig. 1. Jini architecture.

- Being independent of the underlying network infrastructure makes the service architecture extendable.
- MAs allow new services to be provided dynamically either by customization or (re)configuration of existing services.
- MAs provide an effective way for deployment and utilization of advanced services within a distributed environment.

The MA paradigm and emerging agent technologies are considered key for implementing open, flexible and scalable services. There are many commercial and nearly commercial agent platforms, such as Grasshopper (IKV++), Aglets Workshop (IBM), Voyager (ObjectSpace), Concordia (Mitsubishi) and Odyssey (General Magic). Interoperability is therefore becoming an important issue. Sound standards are needed. The Mobile Agent System Interoperability Facility (MASIF) specification by OMG represents the first effort for standardizing agent platforms. It is a specification of an agent framework to support agent mobility based upon the use of the Common Object Request Broker Architecture (CORBA). Although the MASIF standard stresses language independence, it is interesting to note that most notable mobile agent frameworks are implemented in Java. It is our belief that this trend will intensify. For this work we chose Grasshopper as the platform, mainly because of its adherence to the MASIF standard.

In April 1997, CLIMATE—The Cluster for Intelligent Mobile Agents for Telecommunication Environments, a pool of projects within the European Union collaborative research and development program on Advanced Communications Technologies and Services (ACTS), was launched to explore the usage of agent technologies. Most of these projects are located within the Service Engineering, Security, and Communications Management domains. CLIMATE is taking an active part in contributing to relevant agent standards (e.g. OMG, FIPA) and telecommunication standards (e.g. IN, TMN, UMTS standardization). Notably, the Grasshopper MA framework has been developed under the CLIMATE umbrella. A proposal for the use of mobile agents in network management problems has also been forthcoming [20].

2.2. Jini/JavaBeans

The JavaBeans specification is an object-oriented programming interface that is used to build re-useable application or program building blocks called components that can be deployed in a network or any major operating system. Ideally, any Java components conforming to the JavaBeans component model can be reused in any other JavaBean compliant application. Every Bean not only complies with the JavaBean model, it also carries with it all its properties and methods, which can be easily garnered through introspection—a JavaBean property whereby any Bean-aware tool (e.g. a visual programming tool) can analyze and report on how a Bean operates.

Jini [21] technology takes advantage of the Java language. It brings to the network facilities for distributed computing, network based services, seamless expansion, reliable smart devices, and easy administration. It provides lookup services and a network bulletin board (or blackboard) for all services on the network. Jini allows the search of services connected by the communication infrastructure and stores not only pointers to the service on the network, but also the code and/or code pointers for these services. Fig. 1 shows the Jini architecture. The components of the Jini system can be segmented into three categories: infrastructure, programming model and services. The infrastructure is the set of components that enables building of a federated Jini system, while the services are the entities within the federation. The programming model comprises interfaces that enable the construction of reliable services.

2.3. Intelligent networks and service-independent building blocks (SIBs)

Intelligent Network [22–24] services are based on additional service logic and data on top of different switched telecommunication networks. Centralized service nodes known as Service Control Points (SCPs) control the telecommunications network via a dedicated out of band signaling network; i.e. the International Signaling System No. 7 (SS7) network. The bearer switching nodes, known as Service Switching Points (SSPs), provide only basic call processing capabilities. IN service deployment and management is realized through a Service Management System (SMS), which interacts with IN elements via a data communication network. Since the SSPs and the SCP have to interact for each IN service call (usually multiple times), the signaling network and the central SCP may become serious bottlenecks. Furthermore, SCP failures would result in global service unavailability.

The IN platform provides greater flexibility for service creation in general and also for the tailoring of services to suit the exact requirements of a particular customer. IN-based services rely on service-independent building blocks (SIBs) these being the smallest units in service creation. SIBs are reusable and can be chained together in various combinations to realize services. They are defined to be independent of the specific service and technology for which or on which they will be realized.

With more effort to standardize agent platforms, agent platforms are maturing, and with Java as an enabling tool for implementing MAs, the MAs have brought tremendous opportunities for development of MA-based service architectures for IP telephony. It should be noted that MA architectures for traditional IN services in the PSTN [16] and Internet services [25] have already been proposed. The use of CORBA and Java for multimedia services has also been proposed [26] as has the integration of IN and Internet services [27].

3. Mobile agent based advanced service architecture for H.323 supplementary services

3.1. Advanced service architecture

As a consequence of the drawbacks of the existing service architecture defined by H.323 described earlier, we propose an MA-based advanced service architecture for implementing H.323 supplementary services using the widely accepted service provisioning basis (IN), enabling technology (MA, Jini/JavaBeans) and the requirements described in Ref. [11].

The main idea of this architecture is to provision H.323 supplementary services in a uniform, but very flexible way, supporting dynamic deployment of services. Mobile agent platforms are introduced into the devices that are connected to the enterprise LAN. H.323 supplementary services are realized by means of mobile service agents. The key to this approach is to deploy service agents to the service users; i.e. the call parties, which makes this service architecture open, distributed and flexible.

This service architecture allows for open service creation. Supplementary services can be created by a different Service Component Creator (SCC) using Service Components from a Service Component Repository (SCR). Service utilization is realized by activating a caller's User Service Agent (USA) and ultimately activating the callee's USA. A Call Agent (CA) will be instantiated by the result of a USA creating a new CA. Using terminology from the Design Patterns community, the USA represents a factory for call agents. The USA has responsibility for service management. The CA embodies the call processing functionality required to set up the call, e.g. it interfaces with an H.323 protocol stack for call setup. In this way we have clear separation of service management and call processing responsibilities.

This architecture supports universal access to the service through the Jini Lookup process. Service customization is also supported by this service architecture. Each user connected to the network can define his or her own service data. An example of such data could be the number to which calls might be forwarded in a forwarding service like CFU. Service logic is not embedded in the network nodes but ultimately in the end user's terminal. This makes this service architecture highly distributed.

As illustrated in Fig. 2, H.323 gatekeepers and H.323 terminals (users) are connected to the Enterprise LAN. An agency that provides an agent execution environment is attached to each gatekeeper and terminal. User terminals join one gatekeeper's zone through gatekeeper discovery and an endpoint registration process. Lookup Services (LUSs) are more like a blackboard where all the available services' proxy code (interface of a service not the actual service code) is placed. A Service Component Creator (SCC) is responsible for creating components that are made available to an Enterprise Service Creator (ESC) and advertising its services on a LUS, where all the service components are stored in the Service Component Repository (SCR). These service components can be assembled into new services by an SCC; end users can not subscribe to services from an SCC directly. The SCC and SCR bring opportunities for third party service creators and providers, making them able to compete in the service market. LUSs can be local or remote; they are linked by the Internet. A Service Management Unit (SMU) is a device that can be placed in the gatekeeper, or completely separated from the gatekeeper. It manages service subscription using protocols not defined by H.323; e.g. HTTP. If a service is to be dynamically upgraded, the SMU would be involved. Also, the SMU could be involved in ongoing network management of the services. A SMU can discover an enterprise LUS using a multicast protocol; a unicast protocol is used to discover remote LUSs that are outside of the enterprise

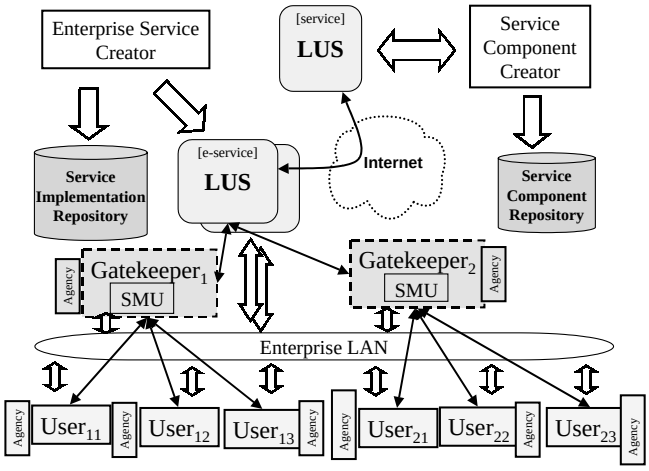


Fig. 2. Mobile code for IP telephony.

LAN. In the latter case, the SMU has to know where the LUS is before it sends out a request to the remote LUS. An Enterprise Service Creator (ESC) is responsible for customizing/assembling the service components into services they provide to the end user using the available code from a Service Implementation Repository (SIR).

In summarizing the roles of the participant components introduced above, Jini is used to provide access to service-related code and service creators generate service components which they advertise by storing them in a Jini lookup service. These components are discovered by Enterprise service creators who generate value-added services specific to their own enterprise needs. These value-added service components are made available to users by advertising them in Jini lookup services visible from inside the enterprise network. When subscribing to specific services, users may ask to be notified of changes to that service. For example, should a service need to be taken off line for maintenance purposes, the users subscribing to this service can be notified thereby making possible the identification of an alternate provider. This notification of service changes allows for enhanced user-controlled service management.

In the following discussions, we will present one scenario for service subscription when only one gatekeeper is involved and one scenario for service utilization when one gatekeeper is involved, and at the same time we will explain all the related components' functionality. These scenarios are presented as they represent familiar H.450.x standard services. However, the architecture proposed in this paper is considerably more flexible. For example, none of the examples presented in this paper require the generation of more than a single CA. Future services will, in all likelihood, require more.

3.2. Service subscription and utilization using mobile agents

There are four phases in the life cycle of a service; they are: service creation, subscription, utilization and withdrawal. Service subscription and utilization using this MA based advanced service architecture are described in this section. The example of VPN service will be used to illustrate the key points in Fig. 3. A VPN can be thought of as a service comprising feature sets that have been customized for a particular enterprise.

Referring to Fig. 3, service subscription consists of four steps, the first of which is shown in the figure. The steps are:

- First, as shown in the figure, the Service Component Creator advertises a generic VPN service proxy object using the Service LUS.
- Next, the Enterprise Service Creator discovers the Service LUS, downloads the VPN service proxy object to its machine, and uses its graphical user interface (GUI) to customize the VPN service for its enterprise users.
- Then, the Enterprise Service Creator uploads its customized service proxy object to the e-Service LUS. This service proxy will be downloaded to the Service Management Unit.
- Finally, any user can send a request to its SMU to subscribe to services, as shown by the top arrow in Fig. 4-1.

Referring to Fig. 4-1, after the SMU receives a service subscription request from the end user, it multicasts the request in the network, discovering LUSs which have supplementary services. Following discovery, the SMU gets a response from the LUS listing all the supplementary services it has on the network and the addresses/URLs of other LUSs outside of the enterprise which have the same kind of services available. Using this list, the SMU constructs a service subscription form and sends it to the end user stating that these are the supplementary services available. The end user selects the services that he wants, fills in the form and sends it back to the SMU. The form includes facilities for the user to specify service related data. The user may not find all the

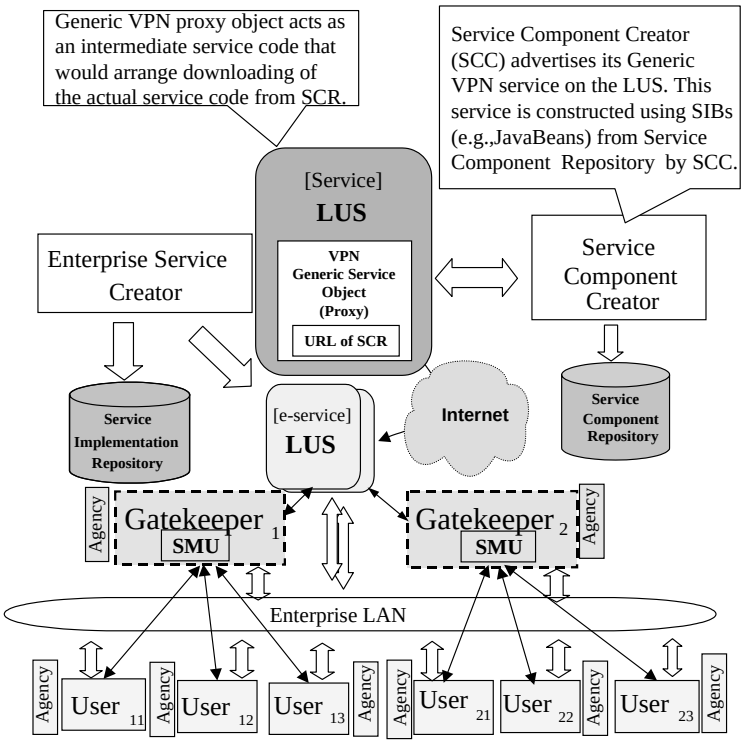


Fig. 3. Service subscription Using Jini.

services he wants. In this case, the user should send a message to the SMU indicating that he/she wants particular services that are not in the form, the SMU then queries other available LUSs external to the enterprise LAN using the addresses/URLs from the last query. After the SMU receives the completed form from the user, it checks its User Profile, which contains all the services that are already in use by each user. Then it sends a request to the lookup service to download the proxy code (interfaces) of the services to which this user has subscribed. When the SMU checks the user profile, the following things may occur:

- The service/services that the user wants to subscribe to is already there; then the SMU sends a notification to the user indicating that the requested service or services are already provisioned.
- Some of the services the user requested are already available to the user; then the SMU sends a notification to the user, and it also sends the proper request to the LUS.
- If all the services are new to the user, the SMU sends a request to the lookup service to download the proxy code of the services that the user has requested.

The service proxy code will be downloaded to the SMU. The proxy code contains the interfaces that a gatekeeper needs to construct a USA, and the location of a SIR where to find the actual service code. There may be many URLs for the addresses of multiple SIRs. A customized USA will be sent to the user agency (as shown in Fig. 4-2), and resides

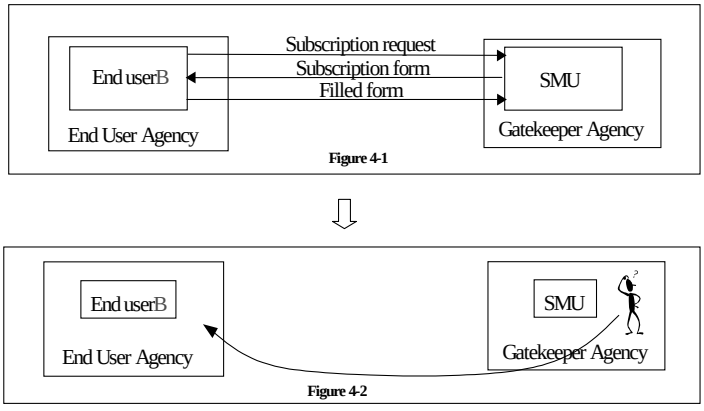


Fig. 4. Service subscription and USA allocation.

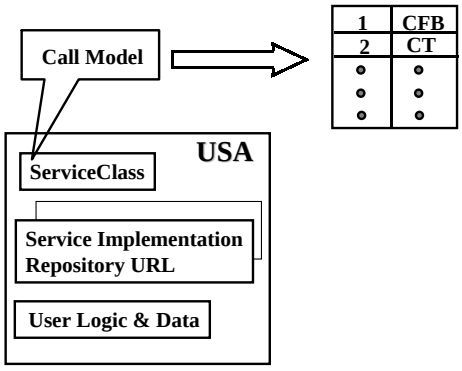


Fig. 5. User service agent.

in the user's terminal. Once the user receives the USA, it acknowledges the SMU.

As illustrated in Fig. 5, a USA consists of ServiceClass, Code Repository URL and User Logic and data. It defines how a call will be processed; e.g. the management of feature ordering. It handles service management and other aspects of network management; e.g. fault management.

3.2.1. More detailed explanation of USA and CA

The Call Model component of the USA is an IN call model consisting of two separate sets of call processing logic: Originating and Terminating call models. The Originating call processing logic provides support to the Calling Party, and is modeled by the Originating Basic Call Model (O-BCSM). The Terminating call processing logic provides support to the Called Party, and is modeled by the Terminating Basic Call Model (T-SCSM).

The call model provides support for a finite state machine with points of interaction with advanced service implementations. In the traditional IN view of advanced services, these points of interaction would be implemented using trigger points. In the approach proposed here, using component-based technology, we would expect JavaBeans to be used with well-known interfaces and the interaction mode would be via method call.

Referring again to Fig. 5, the ServiceClass (Call Model) is a call model specific to an end user. We view the Call Model as a basic service, nothing more.

The User Logic in Fig. 5 represents processing that is required for subscribed services. For a specific service, in one call processing state, it specifies how we deal with this specific service and what the next step is in the call processing. For sophisticated services, the Call Model may have provision for the user to write scripts (rules, perhaps) that add a degree of intelligence to the service. Choosing an example from the e-mail domain, we might choose to write scripts that filter out particular callers, or calls from a specific set of network addresses.

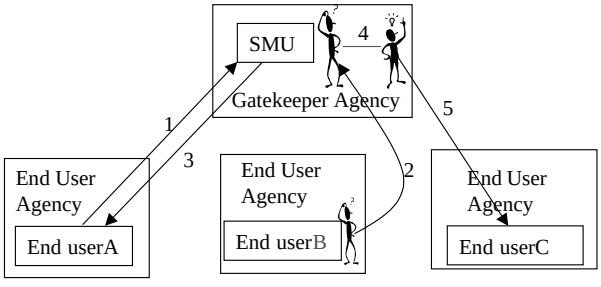


Fig. 6. Call setup using USA.

The User Data in the above figure is the service-related data. For example, after a user chooses a service (e.g. CFU), he/she will also be asked to fill in the phone numbers to which he/she would like to forward calls.

To reiterate, the USA consists of a Call Model, one or more Service Implementation Repository URLs (references to where the service code can be found) and User Logic and User Data.

The Call Agent is an MA that implements the specific call model for a particular end user and makes use of basic call processing functions to control the call setup.

A USA is constructed when the end user sends a request for service subscription. The call model will be unique to the user according to its subscribed services at subscription time. For example, user B may subscribe to Call Forwarding Unconditional (CFU) and Call Transfer (CT). In this case, the ServiceClass component of the USA will be a call model that has different trigger points at different points in the call. For CT, the service trigger point would be in the originating call model. For CFU, the service trigger point would be in the terminating call model. Once constructed, the USA moves to the user local agency from the gatekeeper as shown in Fig. 4-2.

The sequence of how a call is set up using the USA and CA is illustrated in Fig. 6. When user A, who has not subscribed to any advanced services, starts a new call, the Basic Call Processing (BCP) function in the user terminal will be invoked. As a result of the function call, a setup message is sent to the called party (user B) via the gatekeeper (1). If the gatekeeper routed call signaling mode is used, the USA moves to the gatekeeper agency (2). A call signaling message callProceeding is sent back (3) to userA. The SMU checks its user profile—user B has subscribed to CFU and CT—and hence there must be a USA for user B. So user B's USA is activated and a Call Agent (CA) which implements the call model is instantiated (4). The USA instantiates a CA by performing a new Call Agent (USA) call. At this time, the USA and CA reside in the same agency. The CA gets a handle of its parent USA so that these two can communicate with each other by method calls. The CA also makes use of the BCP functions available in the gatekeeper and can also obtain the code it needs (i.e. code related to advanced services) through URLs provided by USA. Then, the CA will take over the call processing, i.e. it acts on behalf of user B who has subscribed to the various advanced services (5).

Call control messages are sent between user A and the CA

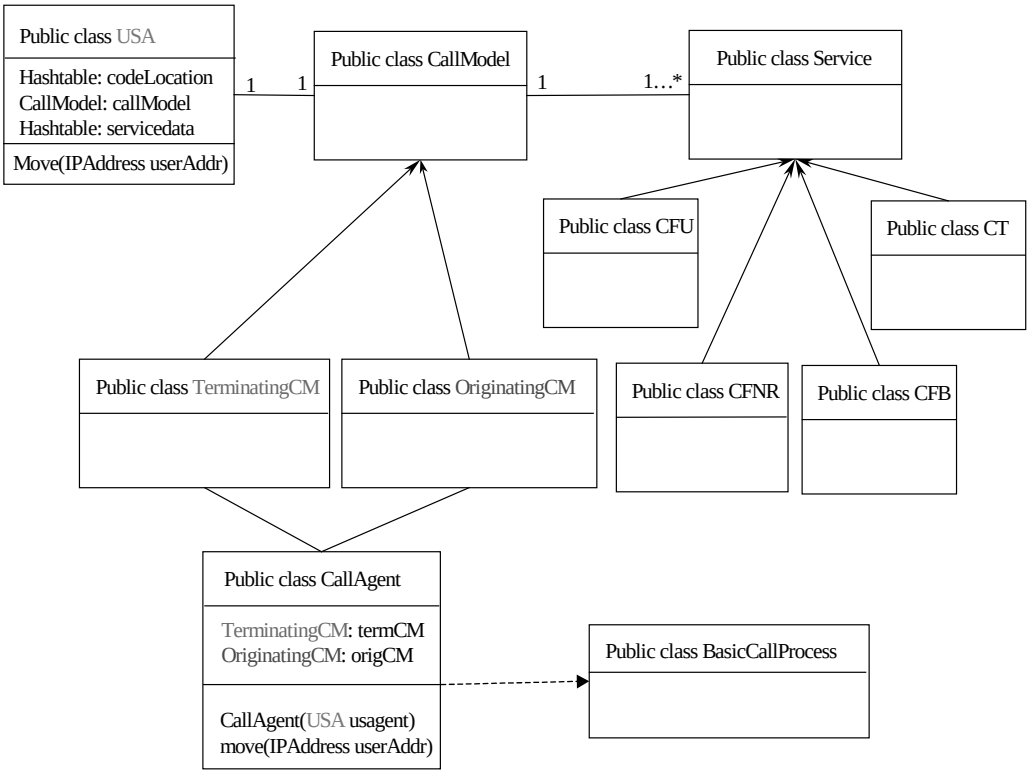


Fig. 7. High-level view of USA–CA interaction.

on behalf of user B. So, the CA will know that whenever it gets an incoming call, it will forward the call the phone number (IP address) that B has specified during the service subscription process.

Fig. 7 illustrates a high-level class interaction view of the USA and CA. In the figure, the attribute codeLocation of the class USA is a hashtable which contains associations of serviceName and codeLocation URL, one URL for each service as different services may come from different vendors.

Service Utilization is shown in Fig. 8. Once a user has initiated a call (1), the USA moves to the gatekeeper's agency if it is gatekeeper routed call signaling, and a Call Agent (CA) is instantiated using the code retrieved from the SIR. The CA sends call signaling to the called endpoint via the gatekeeper.

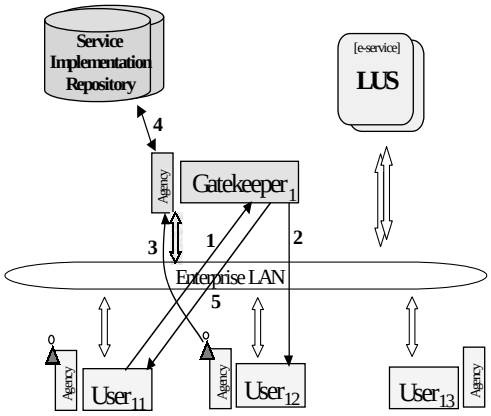


Fig. 8. Service utilization.

For the called endpoint USA, as soon as it receives the first setup message, the USA moves to the gatekeeper if it is a gatekeeper routed call signaling, a CA is instantiated using the code retrieved from the SIR, and a CA sends all call signaling to the calling endpoint via the gatekeeper. For endpoint-to-endpoint call signaling, the USA will be

Table 1
Service requirement support

1. Support all life cycles phases	Most supported
2. Support a wide range of services	Yes
3. Support multi-player environments	Through JavaBeans
4. Rapid service creation and deployment	Yes, through service creation environments (vendor and enterprise)
5. Tailored services	Yes, through enterprise service creation environment and end-user customization
6. Independent evolution of network and service infrastructure	Not really, still have to interwork with classical telephony network
7. Universal access	Yes, Jini provides location transparency
8. Interwork with other service architecture	Possible with IN if gateway technology built

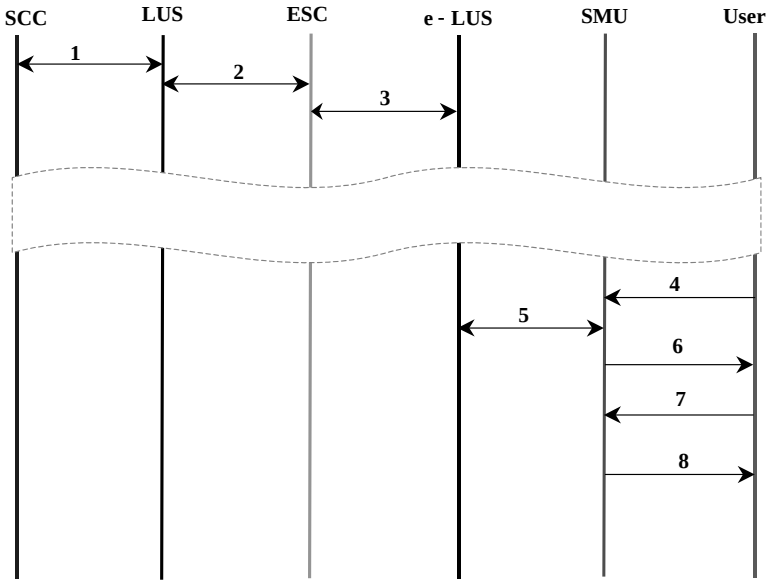


Fig. 9. VPN service subscription.

executed in the user local agency instead of moving to the gatekeeper's agency.

3.3. Architecture discussion

The service requirements from Ref. [11] are listed in Table 1. The intention with the architecture proposed in this paper is to meet more of the needs of the phases of the lifecycle of a service. Clearly, this mobile agent advanced service architecture fulfills more of the requirements for today and tomorrow's IP Telephony advanced service architecture when compared to traditional service creation and utilization environments such as are found in PSTN networks (for example).

4. Service subscription and utilization scenarios

In this section, four advanced service scenarios are presented. One scenario is VPN service subscription. Here, a VPN is a private enterprise telephone network established by the enterprise administrator. Such a VPN can be used, for example, to control who has access to VPN services such as long-distance calling etc. The remaining three are used to illustrate the USA's and CA's movement on deployed supplementary services using H.323 messaging. Several examples are provided with one gatekeeper involved and gatekeeper routed call signaling; i.e., Call Forwarding (CF) in Section 4.2, Call Transfer (CT) in Section 4.3, and VPN in Section 4.4. CF and CT are defined by ITU-T standard H.450.x; VPN is not currently defined in the standards.

4.1. VPN service subscription

Fig. 9 is explained by the following.

Assumption: The messages used in Fig. 9 are not defined by H.323, they need to be defined in the future.

Description:

1. The SCC discovers a LUS and advertises the generic VPN on the LUS by uploading the generic VPN service proxy object.
2. The ESC discovers the generic VPN and downloads the VPN service proxy object.
3. After the ESC has customized the generic VPN service for its enterprise users, it discovers and uploads the customized VPN to the enterprise LUS, the e-LUS.
4. The end user requests the use of the customized VPN service.
5. The SMU discovers the e-LUS that has the VPN service and downloads the service proxy object.
6. The SMU uses the downloaded proxy object to construct a USA, and sends it to the user agency.
7. When the user receives a USA, it sends an acknowledgement to the SMU indicating that a USA has been received.
8. The SMU sends a message to the end user indicating that service subscription is completed.

4.2. Call forwarding

4.2.1. Call forwarding unconditional using end-to-end call signaling

Fig. 10 is explained as follows.

Assumption: The client application will be responsible for sending messages specified by the CA to the other applications which are dealing with the call setup process. The CA can accomplish this by calling a client application's interfaces.

Description:

1. An Originator (Caller) application sends a SETUP

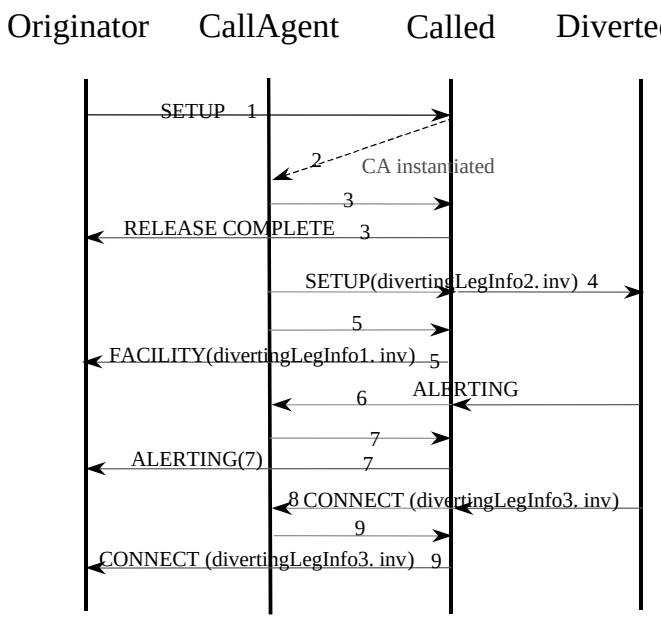


Fig. 10. CFU (end-to-end call signaling).

message to the Called party (Callee), a flag will be set in SETUP message's NonStandardControl field to activate the USA, so that a Call Agent is instantiated.
2. After the call agent is instantiated—it resides in the callee's user agency—it takes over the call processing from the client application, sending out all the call setup related call signaling messages.
3. The Call Agent sends RELEASE COMPLETE to the caller.
4. The Call Agent sends SETUP with divertingLegInfo2.inv to the diverted-to party.
5. The Call Agent sends FACILITY with divertingLegInfo1.inv to the calling party.
6. The Diverted-to party sends ALERTING to the Call Agent.
7. The Call Agent sends ALERTING to the calling party.
8. The Diverted-to party sends CONNECT to the Call Agent.
9. The Call Agent sends CONNECT to the calling party.

4.2.2. Call forwarding unconditional using gatekeeper routed call signaling

Fig. 11 is described as follows.

Assumption: Same as CFU using end-to-end call signaling.

Description:

1. The originator sends SETUP message to the Gatekeeper.
2. The Gatekeeper responses with CALL PROCEEDING.
3. The Gatekeeper sends SETUP with diveringLegInfo4.inv to called endpoint.
4. The called endpoint sends RELEASE COMPLETE with USA in the NonStandardData field.
5. Once the Gatekeeper receives the USA, a Call Agent is instantiated.
6. The Call Agent sends SETUP with divertingLegInfo2.inv to the diverted-to endpoint.
7. The Call Agent sends FACILITY with divertingLegInfo1.inv to the calling endpoint.
8. The Diverted-to endpoint sends ARQ to the Gatekeeper indicating it will accept the call.
9. The Gatekeeper sends ACF to the Diverted-to endpoint with the Gatekeeper's call signaling transport address.
10. The Diverted-to endpoint sends ALERTING to the Call Agent via the Gatekeeper.
11. The Call Agent sends ALERTING to the originator.
12. The Diverted-to endpoint sends CONNECT with divertingLegInfo3.inv to the Call Agent via the Gatekeeper.
13. The Call Agent sends CONNECT with divertingLegInfo3.inv to Originator.

4.3. Call transfer

Fig. 12 is explained by the following.

Assumption: Call Agent has been instantiated in the caller agency when the caller started making call.

Description:

1. The Transferring endpoint (A) sends FACILITY with CTInitiate.inv to CA.
2. The Call Agent sends SETUP with CTSetup.inv,opt.CTUpdate.inv to the transferred-to endpoint(C).
3. The Call agent sends CONNECT with CTSetup.rr,opt.CTUpdate.inv to the transferred-to endpoint(C).
4. The Call Agent sends FACILITY with CTComplete.inv to the transferring endpoint(A).
5. The Call Agent sends Terminal Capability Set C to the transferred-to endpoint.

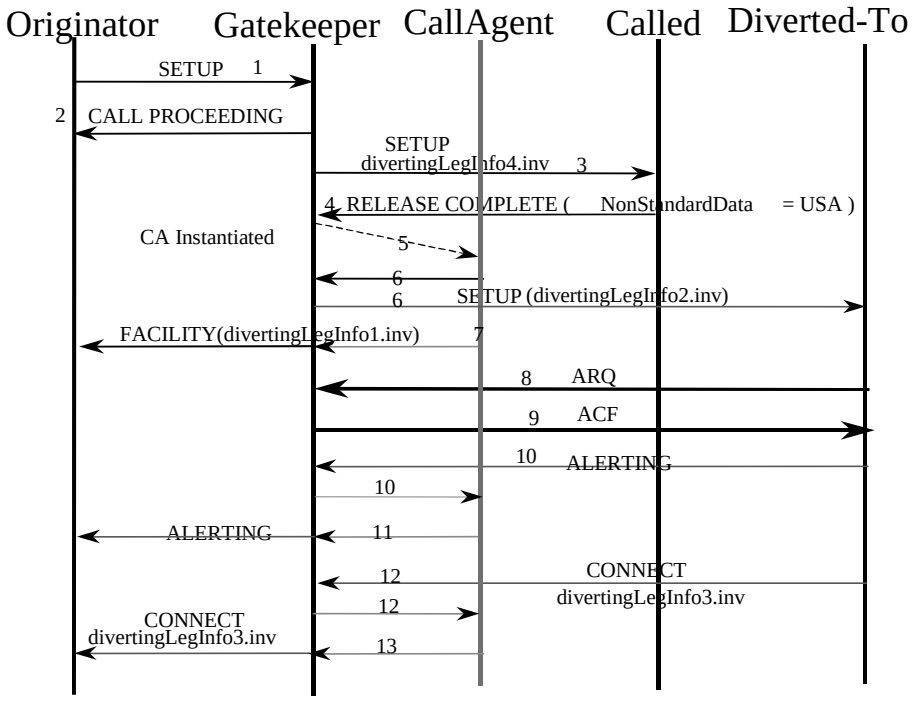


Fig. 11. CFU (Gatekeeper routed call signaling).

6. The transferred-to(C) endpoint sends TCS = 0 to the Call Agent.
7. The Call Agent sends TCS = 0 to the transferring(A) endpoint.
8. The Call Agent sends TCS = 0 to the transferred (B) endpoint.
9. Close Channels between transferring point (A) and transferred endpoint (B).

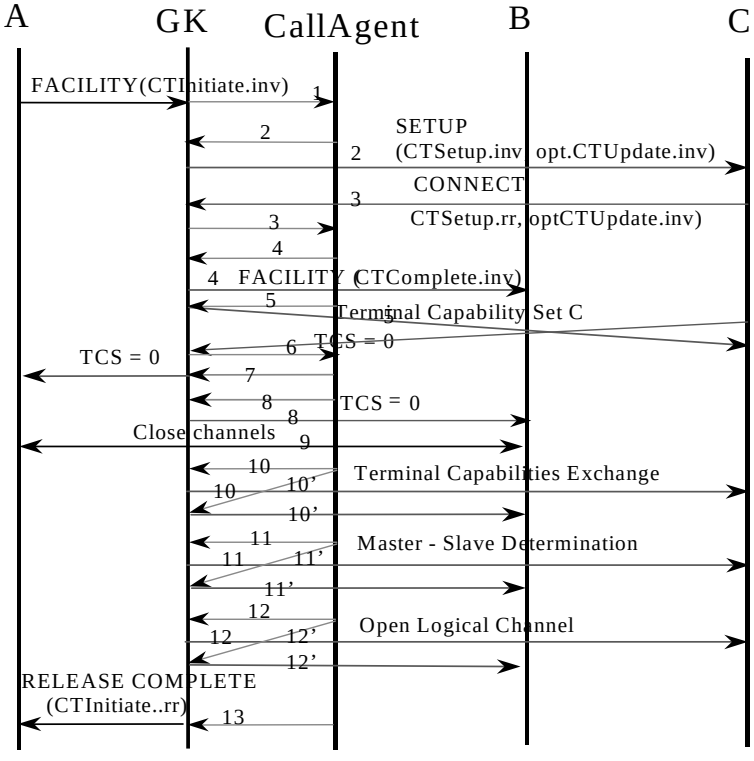


Fig. 12. CT (Gatekeeper routed call signaling).

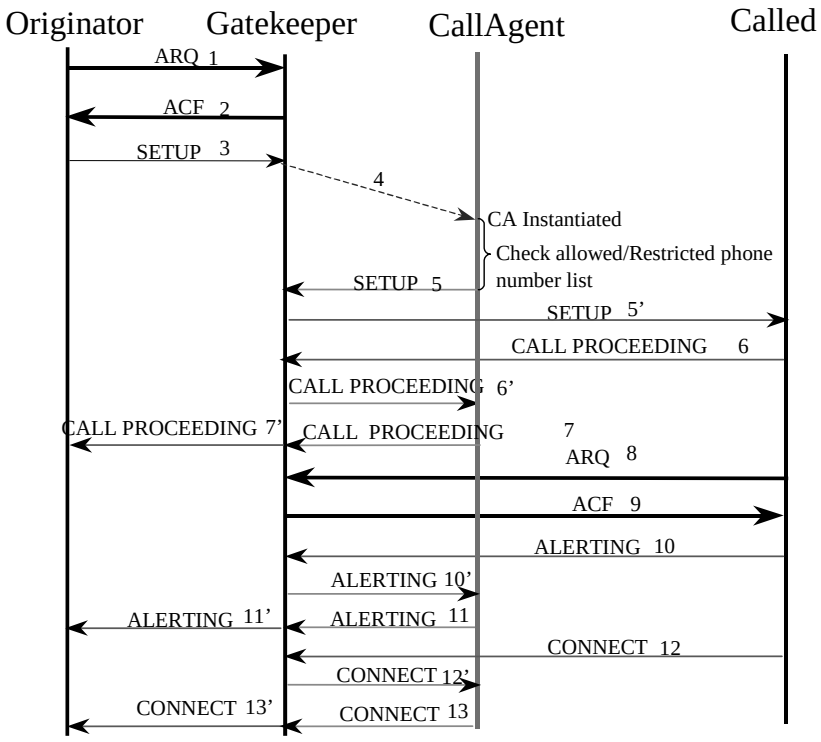


Fig. 13. OGA/OGR (Gatekeeper routed call signaling).

10, 10′. CA sends Terminal Capability Set to the transferred (B) and transferred-to(C) endpoints.
11, 11′. CA sends Master/Slave Determination to the transferred(B) and transferred-to(C) endpoints.
12, 12′. The Call Agent sends Open Logic Channel to the transferred (B) and transferred-to (C) endpoints.
13. The Call Agent sends RELEASE COMPLETE to the transferring (A) endpoint.

4.4. VPN—outgoing call allowance (OCA)/outgoing call restriction (OGR)

The OCA and OGR VPN features allow an enterprise administrator to restrict the numbers that may be called from within an enterprise. For example, an easily accessible terminal might be restricted to numbers within the enterprise whereas a terminal on an employees desk might have unrestricted dialing privileges. Fig. 13 is explained in the following.

Assumption: Since the VPN service is not defined by H.323, here we are using the H.225.0 call signaling in order to illustrate the call management sequence using a Call Agent, and also to make it easier to understand by using the same style of call sequence diagram. The messages used here need to be identified in the future.

Description:

1, 2. The originator and its gatekeeper exchange admission messages.
3, 4. The originator (Caller) application sends a SETUP message to the Called party and a flag will be set in the SETUP message's NonStandardControl field to activate the USA residing in the gatekeeper, so that a Call Agent is instantiated. After the call agent is instantiated, it resides in the gatekeeper's agency. It checks the allowed/restricted phone number list first. If the called number is in the allowed phone number list or not in the restricted phone number list, then the CA takes over the call processing from the client application, sending out all the call setup related call signaling messages. (If the called number is denied, then the CA will send CALL RELEASE to the originator).
5. 5′. The Call Agent sends a SETUP message to the called party via the gatekeeper.
6, 6′. The Called party sends a CALL PROCEEDING message to the CA via a gatekeeper.
7, 7′. The CA sends a CALL PROCEEDING message to the Originator via the gatekeeper.
8, 9. The Called party and its gatekeeper exchange admission messages—ARQ, ACF.
10, 10′, 11, 11′. The Called party sends an ALERTING message to the CA via the gatekeeper, and the CA sends an ALERTING message to the originator via the gatekeeper.
12, 12′, 13, 13′. The Called party sends a CONNECT message to the CA via the gatekeeper, and the CA sends a CONNECT message to the originator via the gatekeeper.

5. Conclusions

The mobile agent based advanced service architecture solution proposed in this paper provides the following features and benefits. The architecture can:

- Enable the provision of flexible software solutions, where H.323 supplementary services software is partitioned into mobile service agents realizing dedicated functionalities (e.g. IN service features).
- It enables on demand provision of customized supplementary services by dynamic construction of a user service agent that uses downloaded service code from the SEC or ESC to the gatekeeper.
- It allows for decentralized realization of supplementary services, by means of bringing the user service agents directly onto the user terminals.

We have demonstrated that mobile agents may be successfully integrated with H.323 IP telephony protocols for the provision of advanced services. The architecture that this paper proposes seeks to address the entire service lifecycle, an important consideration in opening the IP telephony marketplace to non-traditional telephony service providers.

Our future work consists of the construction of IP telephony services not currently defined by the existing H.450.x specifications in order to further validate the architecture. We are also currently evaluating the traditional IN SIBs with a view to re-factoring the behavior provided by them. Finally, a performance evaluation of the existing architecture using typical hardware and software platforms needs to be performed.

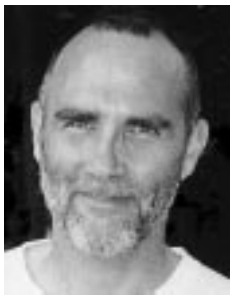
Result of these activities will be communicated in future publications.

References

[1] A. Gary, H.323: the multimedia communications standard for local area networks, IEEE Communications Magazine December (1996) 000.
[2] H.323 Tutorial; http://www.webproforum.com/trillium/index.html.
[3] Internet Telephony; http://www.webproforum.com/siemens2.
[4] ITU-T Rec. H.225.0, Media Stream Packetization and Synchronization for Visual Telephone Systems on Non-guaranteed Quality of Service LANs, 1997.
[5] D. Clark, A Taxonomy of Internet Telephony Applications; http://itel.mit.edu/itel/publications.html.
[6] ITU-T Rec. H.323, Visual Telephone Systems and Terminal Equipment for Local Area Networks which Provide a Non-guaranteed Quality of Service, 1996.
[7] H. Schulzrinne, J. Rosenberg, Comparison of H.323 and SIP; http://www.cs.columbia.edu/~hgs/sip/h323.html.
[8] B. Pagurek, T. White, A quick evaluation of H.323/H.450, Technical Report SCE-99-02, Systems and Computer Engineering, Carleton University, April 1999.
[9] The Problems and Pitfalls of Getting H.323 Safely Through Firewalls; http://support.intel.com/support/videophone/trial21/H323_WPR.HTM.
[10] R. Minetti, E. Utsunomiya, The TINA service architecture; http://www.tinac.com/specifications/abstract.htm.
[11] R.H. Glitho, Advanced services architectures for Internet telephony: State of the Art and Prospects, Ericsson Research Canada Technical Report, Montreal.
[12] T. Magedanz, K. Rothermel, S. Krause, Intelligent agents: an emerging technology for next generation telecommunications? INFOCOM'96, San Francisco, USA, 1996.
[13] J. Kiniry, D. Aimmerman, A Hands-on look at Java mobile agents; http://computer.org/internet/ic1997/w4021abs.htm.
[14] H.S. Nwana, Software agents: an overview, Knowledge Engineering Review 11 (3) (1996) 1–40.
[15] T. Magedanz, K. Rothermel, S. Krause, Intelligent agents: an emerging technology for next generation telecommunications? IEEE INFOCOM, San Francisco, April 1998.
[16] M. Breugst, T. Magedanz, Mobile agent—enabling technology for active intelligent network implementation, IEEE Network May/June (1998) 000.
[17] D.B. Lange, Present and future trends of mobile agent technology, Second International Workshop on Mobile Agents'98 (MA'98) Stuttgart, Germany, September 1998.
[18] A. Fuggetta, G.P. Picco, G. Vigna, Understanding code mobility, IEEE Transactions on Software Engineering 24 (1998) 000.
[19] D. Chess, C. Harrison, A. Kershenbaum, Mobile agents: are they a good idea?—update, Mobile Object Systems: Towards the Programmable Internet, Lecture Notes in Computer Science, 1222, Springer, Berlin, 1997 pp. 46–48.
[20] A. Bieszczad, B. Pagurek, T. White, Mobile agents for network management, IEEE Communications Surveys September (1998) 000.
[21] Jini specifications; http://www.sun.com/jini/specs.
[22] T. Magedanz, Intelligent Networks, International Thomas Computer Press, 1996.
[23] T. Jan, Intelligent Networks, Artech House, 1994.
[24] V. Avery, J. Matta, Intelligent Networks: A Concept for the 21st Century; http://www-dse.doc.ic.ac.uk.
[25] A. Park, A. Kupper, S. Leuker, JAE: A multi-agent system with Internet services access, Proceedings of the Fourth International Conference on Intelligence in Services and Networks, IS&N'97 Cernobbio, Italy, May 1997.
[26] A. Limongiello, R. Melen, M. Rocuaao, V. Trecordi, J. Wojtowicz, An experimental open architecture to support multimedia services based on CORBA, Java and WWW technologies, Proceedings of the Fourth International Conference on Intelligence in Services and Networks, IS&N'97 Cernobbio, Italy, May 1997.
[27] O. Miauno, J. Urata, Y. Sueda, Y. Niitsu, Advanced intelligent network and the Internet combination service and its customization, IEICE Transactions on Communication E81-B (8) (1998) 000.

Prof. Bernie Pagurek received his PhD in Electrical Engineering from the University of Toronto and is currently with the Department of Systems and Computer Engineering at Carleton University in Ottawa, Canada. His research interests include Communication Network Management, Mobile Agents, Fault Diagnosis and Alarm Correlation in Networks, and Network Service Configuration,. He is the principal investigator of a CITO (Communications and Information Technology Ontario) research project on Network Management.

Jingrong Tang received a BEngng degree in Telecommunications Engineering from Changchun Post&Telecommunication Institute (1990), China. She has worked as a telecommunication engineer in Beijing Telecom for three years. She is currently pursuing her MEngng degree in Systems and Computer Engineering at Carleton University, Canada. Her research interests include Mobile Agents and IP telephony.



Tony White received BA and MA degrees in Theoretical Physics from Cambridge University in 1978 and 1982, respectively. He has a Master of Computer Science from Carleton University, awarded in 1993 and is currently studying for a PhD in Systems and Computer Engineering there. Tony has published papers in the areas of Genetic Algorithms, Object Oriented Design, Mobile Agents, Biologically-inspired Agents, Expert Systems and Fault Diagnosis and has co-authored six patents in the Telecommunications domain. His research interests center on the exploitation of biological metaphors for problem solving in Telecommunications, this being the subject of his PhD research.

Roch H. Giltho works for Ericsson Research in Montreal, Canada where he leads research activities in service engineering for Internet Telephony. He joined Ericsson Research, Canada in 1993 after having worked three years for Ericsson Telecom in Stockholm, Canada. Prior to that he worked five years for a computer manufacturer in Oslo, Norway. He holds MSc degrees in Business Economics (University of Grenoble, France), Pure Mathematics (University of Geneva, Switzerland) and Computer Science (University of Geneva). He is the Editor-in-Chief of IEEE Communications Surveys (http://www.comsoc.org/pubs/surveys/).