

A Personal Communication Service Creation Model for Internet-based Unified Communication Systems

Helen J. Wang, Ascan Morlang, Randy H. Katz
Computer Science Department
University of California at Berkeley, CA USA
{helenjw, randy}@eecs.berkeley.edu, {morlang}@icsi.berkeley.edu

Abstract—

Advances in the Internet and telecommunications technologies have spurred many research efforts in Internet-based unified communication systems, which integrate heterogeneous devices and networks (PSTN, cellular networks, or the pager networks). Nonetheless, these systems lack support for a systematic, easy, flexible way of customizing and creating services at the system level as well as at the user level. In this paper, we present such a service creation model by exposing a limited set of primitives from a call model that is network and device independent. Our model and framework not only supports the traditional telephony call services easily, but also enables novel ways of creating third party services such as application-specific billing services.

Keywords—

Unified communication systems, Internet telephony, service creation model, call processing, system primitives, preference specification

I. INTRODUCTION

Rapid advances in telecommunications and Internet technologies have enabled powerful means of communications through heterogeneous devices such as telephones, cellular phones, pagers, PCs, and PDAs that access diverse networks such as the Public Switched Telephone Network (PSTN), cellular networks, pager networks and the Internet. Personalized and integrated use of these heterogeneous devices and the services provided by their networks (e.g., redirect all incoming communication to my e-mail when I am in a meeting) is in great demand; and have spurred many research efforts in Internet-based unified communication systems. These systems integrate heterogeneous devices and networks by having both control and data transport of a communication session on top of the Internet, using gateways to interwork with access networks, performing signaling translation, and data stream packetization and transformation. Nonetheless, service introduction and implementation in such systems are still immature in lacking a user-friendly and systematic service creation model. The services in question here refer to communication control and redirection services allowing users to express when they want to be reached on what devices under what circumstances.

In this paper, we present a service creation model and framework that enables arbitrary customization of the basic communication process, and easy third party service creation in a unified communication system. Our basic communication process generalizes the traditional basic telephone call model, which is a two-party telephone communication, to a multi-endpoint, invitation-based communication where multiple heterogeneous endpoints can participate in the same communication session; and the session participation is achieved through invitation from the existing communicating endpoints in the session.

We have identified the following goals and requirements in our design:

- **Simplicity and user-friendliness:** The service creation

process must be simple and user-friendly enough that any user, if they wish, can customize their communication services.

- **Flexibility:** The underlying system must have sufficient mechanisms to support a wide span of services and arbitrary user customization.
- **Extensibility and completeness:** It is difficult to design and implement a system that supports a complete set of services. The design of the underlying communication system must allow systematic and easy extension.
- **Service portability:** The specification for service customization and creation should apply to systems built by different vendors with the same service creation model without any modification.

Our approach to service customization and creation is based on simple, network-independent, extensible state machines for communication establishment and maintenance. This greatly eases new endpoint or network introduction, and reduces the complexity of the call model. We also identified the system parameters and primitives to be exposed from the state machines, and reflected the state machine execution process into a user friendly preference specification process for arbitrary service customization as well as for sophisticated service creation through partial state machine insertion. This systematic way of service customization and creation is also an important contribution of our work. For easy, flexible provisioning of third-party services, we benefit from the basic multi-endpoint communication model by introducing service agents encapsulated as a participating communication endpoint.

As a first step evaluation of our model, we worked through most of the traditional telephony services. We also explored novel implementation of services like Pay-Per-View. Our preliminary results are very encouraging.

For the rest of the paper, we first set context and describe what constitutes a unified communication system to enable flexible, easy and rapid service customization and creation in Section II. Then, we present our service creation model and framework in detail in Section III. We discuss the related work in Section IV and conclude our paper in the last section.

II. UNIFIED COMMUNICATION SYSTEMS

Before describing the service creation model for the unified communication system, we begin with the required capabilities and desirable properties of a unified communication system below, along with our approach in building our test-bed, the ICE-BERG system.

- **Any-to-any communication:** Any-to-any communication refers to the ability to effec-

tively support communication between all types of devices. To enable any-to-any communications, unified communication systems need a component that inter-works with various networks for signaling translation and packetization. In ICEBERG, this component is called an ICEBERG Access Point (IAP). It encapsulates the access network-specific implementation, and serves as a gateway between the access network and ICEBERG. There is one type of IAP per access network.

An ICEBERG Call Agent performs communication session establishment and control on behalf of a participating endpoint. It runs a network and device independent state machine, and is a network independent component. Throughout a session, Call Agents interpret and carry out user specified preferences and services through the interactions with the Preference Registry and Personal Activity Coordinator (explained below).

- **Personal mobility services:** Personal mobility means treating people, rather than devices, as communication endpoints. Every person using ICEBERG has a unique ID. The mapping between the specific device ID and unique ID is done through the Name Mapping Service (NMS). Other communication systems that achieve personal mobility have such a component, too.
- **Communication service customization:** To allow end users to customize their communication service (such as when they want to be called, on what device, under what condition, and by whom), we use a Preference Registry (PR) to store and manage user preferences. We have also seen the equivalent component in other systems.
- **User activity-driven services:** ICEBERG can support a new kind of communication service based on user activity. This type of service generalizes the location-based services that have appeared in many other systems. Instead of customizing the communication service based on the current user location, we allow the current user behavior (such as “I am talking to an important person”) to be tracked and used for customization. The Personal Activity Coordinator (PAC) performs the tracking.
- **Multi-endpoint communication as the Basic service:** All communication systems provide some *basic services* which are the building blocks for additional services. The traditional telephone system provides two party call between two homogeneous devices (namely, telephones). We believe, that in a unified communication system on the Internet, it is easy and beneficial to enable *multi-endpoint communications* as the basic service. Multi-endpoint communications take place among any number of heterogeneous devices or service endpoints. The new building blocks from multi-endpoint communications, namely, inviting a device or quitting during a communication session, make conference calls, service handoff (i.e., switching devices during an active session) and other endpoint changing services (such as call forwarding or transfer) first class services. These endpoint changing services are simply carried out by inviting a new endpoint, and then leaving the communication session.

For more details on the ICEBERG architecture, please refer to [10].

III. THE SERVICE CREATION MODEL AND FRAMEWORK

Given the capabilities of a unified communication system as described in the previous section, we now present how to achieve communication service customization and creation in such a system.

In our model, there are three ways of customizing or creating services:

1. Customize the communication service through simple *preference specification*. For example, services like “direct all incoming communications to my e-mails after work” can be easily specified through condition-action pairs (Section III-B);
2. Create sophisticated communication service through *extending the basic call state machine* by inserting a partial state machine into the customizable event handlers. Services like “if my cell phone is not reachable, try my office phone, if my office phone is not reachable, try my home phone, ...” need to be implemented through state machine insertion (Section III-C);
3. By having multi-endpoint communication as the basic service of the communication system, it enables novel ways of constructing sophisticated services such as billing and authentication as third party services in a much simplified fashion. The basic technique is to encapsulate the billing or authentication agent as an endpoint participating the multi-endpoint communication session (Section III-D).

We will first describe the system interface exposed by the underlying communication system (Section III-A). Then, we explain how services are carried out in the three ways above in detail.

A. System Interface

The system exposes various parameters for communication customization, like time-of-day, callee’s current activity, caller ID or terminal address. Users use these parameters to indicate the conditions under which certain actions should take place (for example, when caller is Helen, direct her call to my laptop phone program). The exposed parameters are listed in table I. This parameter, “current activity”, indicates the current activity of the user, and is maintained by the activity tracking component of the system (i.e., the Person Activity Coordinator in ICEBERG). The actions are a composition of system primitives which are shown in table II.

Information Type	Parameters
<i>Call related information</i>	caller ID, callee ID, caller format additional call information
<i>User information</i>	current activity, caller’s location (if available), callee’s location (if available), capabilities of current location
<i>General information</i>	time of day

TABLE I
SYSTEM PARAMETER FOR CUSTOMIZATION

To enable extending the basic state machines, the system provides the mechanisms to allow new variables, state machine

System primitives
Invite an endpoint
Invite a new user
Send data object
Out-of-band signaling to endpoints (e.g., hang-up)

TABLE II
EXPOSED SYSTEM PRIMITIVES

events, and handlers to be registered.

Next, we explain how users specify their preferences using these system parameters and primitives in detail.

B. Preference Specification

User preferences take the form of condition-action pairs, and are specified with a user-friendly GUI. Conditions are conjunctions of the boolean expressions formed by system parameters (shown in Table I). Actions are sequences of system primitives (shown in Table II). User preferences are carried out on the Call Agents. They are retrieved from the Preference Registry at the instantiation time of the Call Agent, and interpreted during the state machine execution on the Call Agent.

We observe that all system parameters can be categorized to ease the process of preference processing. For example, *caller ID* could be categorized into “friends”, “co-worker”, “VIP”, “family” and “default”, *calling time* into “work”, “dinner”, “private” and “default”. Therefore, the conditions in user preferences are mostly set operations (such as if caller ID is a friend). We allow users to name the categories or use the system defaults.

Now, we address some of the issues with preference specification. Users specify preferences off-line (meaning not during a communication session) through a configuration tool or a web service in which the graphical user interface guides user through this specification process. We also plan to support users to specify their preferences through heterogeneous devices.

Feature interaction related to preference specification happens when user specified preferences are ambiguous and in conflict with one another [11]. For example, in telecommunication networks, the subscription of both the voice-mail feature and call forwarding upon no answering feature would cause a feature interaction, as when the telephone is ringing for sometime, it is unclear whether the system should invoke the voice-mail, or forward the call to another endpoint. In traditional telecommunication networks, although mechanisms such as voice-mail and call forwarding exist, the system lacks policy specification tool to glue these mechanisms together nicely. In ICEBERG, the entire communication process is exposed to the user for customization, which is an integrated customization process of the communication service rather than independent, piece-wise service subscription. This way, we can detect the single user feature interaction at the preference specification time. Resolving feature interaction in a user-friendly fashion is a challenging problem which is part of our future work.

C. State Machine Extension

Sophisticated users are allowed to register their own state machine events (expressed as conditions) and event handlers (composition of system primitives) as part of a complicated action,

which is essentially a partial state machine being inserted into the basic state machine.

Our state machine diagrams are presented using the following notation. States are in oval shapes; event handlers are in rectangle shapes. Events that trigger the corresponding event handlers are above the arrows pointing to the event handlers. Different conditions cause the handler to perform different actions and produce different output events on some entity (such as gateways or the remote call agent).

Figure 1 and Figure 2 show the session establishment and control state machines for both the originating and terminating side, respectively. Event labels starting with “EI” indicates the events on IAPs, while “EC” indicates the events on Call Agents. Common events for all states, such as device hang-up or remote Call Agent termination, are omitted in the figures for clarity.

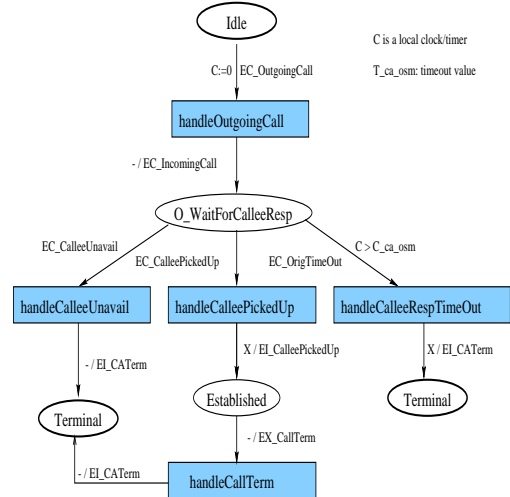


Fig. 1. Originating Call State Machine on Call Agents

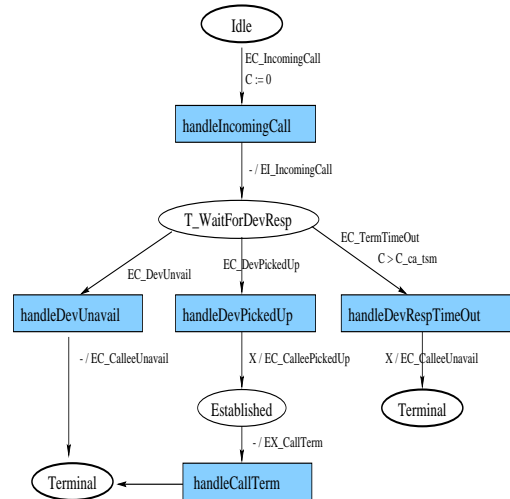


Fig. 2. Terminating Call State Machine on Call Agents

Both the design and implementation of the state machines on

the Call Agents are generic and are designed independent of the networks or endpoints involved in the ICEBERG system. This is in contrast to the basic call systems in the Intelligent Network of the telecommunications systems where the state machines are specifically designed for the PSTN network and telephones. In our system, the device-specific capability is isolated within the IAPs. For example, events like “phone off-hook”, “digits collected” are part of PSTN IAPs designed for telephones only. The IAPs collect these finer-grained events and translate them into generic ones, such as event “EC_OutgoingCall”. This event indicates an out-going call request from an IAP to its Call Agent, encapsulating a sequence of network-specific information collecting events and contains information such as caller and callee ID. Network-independence in the state machine design is beneficial because incorporating a new device or a network will only involve a new IAP implementation for that endpoint.

Figure 3 shows an example of extending the basic state machine on the originating Call Agent (dashed box) allowing communication initiation retries when callee is found unavailable.

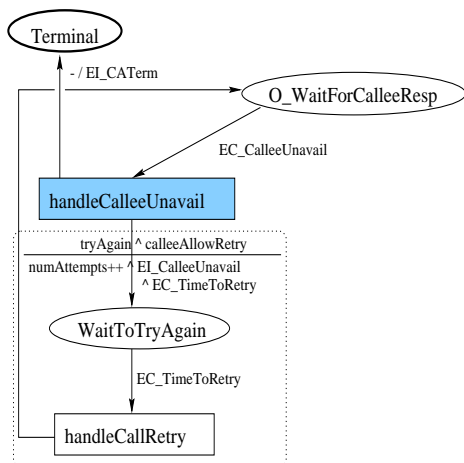


Fig. 3. Example of inserting partial state machine into the basic state machine.

D. Third Party Services

Complex services can be realized by encapsulating service agents behind the IAP interface. The encapsulated service agent participates in the communication session as an endpoint and carries out its functionality. The nice property of this approach is the clean separation of the application-specific function and the basic service functionality.

We illustrate this design by walking through the realization of a typical Pay-Per-View service which includes the access, billing and video distribution agents. These agents are encapsulated behind the IAP interface. At the beginning of a session, a user invites an access agent which validates the user’s identity and retrieves his payment preferences. After this initial session setup, the access agent invites the distribution agent and a billing agent as if they are communication endpoints. The billing agent monitors the session and charges the user accordingly. If either the distribution agent or the billing agent leaves the session, the other agents will also terminate to prevent billing without distribution or uncharged viewing. From this example, we can see

that application-specific billing mechanisms can be easily introduced in this fashion.

E. Service Realization

As a first line evaluation, we evaluate the effectiveness of our service creation model by carrying out Intelligent Network (IN) service features.

Simple preference specification in our system can accomplish many IN features. These features include those used for customized addressing and routing of calls, like *Call Forwarding*, *Call Transfer*, *Originating/Terminating Call Screening*, *Time dependent routing*, *Follow-me Diversion*, and *One-Number*. In the context of ICEBERG, these features are carried out through the interactions of the Call Agent, the Personal Activity Coordinator, and the Name Mapping Service [10]. Since the ICEBERG system inherently supports multi-endpoint communications, IN features for conference calls also become first-class services. These features include: *Multi-Way Calling*, *Consultation Calling*, *Meet-Me Conference*.

Features that require new states in their operations are implemented by inserting a customized partial state machine into the originating or terminating state machines on the call agents. Features belonging to this category are *Call Queuing*, *Call Gapping* and *Call Limiter*, and *Call Distribution*.

Charging features are not directly supported because billing is highly application and business model dependent. Therefore, we believe that it is not suitable to make them as system primitives. Billing services should be realized as separate service agents, as shown in Section III-D.

Other features, which can be realized as encapsulated services, include *User Prompter*, *Customer recorded announcement*, *Call hold with announcement*, *Authentication*, and *Authorization Code*.

IV. RELATED WORK

In this section, we compare and contrast our work with the existing approaches. We first describe the very successful IN service creation model from telecommunications. Then, we evaluate the Internet telephony approach exemplified by the Session Initiation Protocol. Finally, we address JTAPI and Parlay.

A. Intelligent Networks

A desire to more rapidly deploy new services in the telecommunications network has driven the development of the Intelligent Network (IN). This is achieved by creating a standardized service creation environment independent of the underlying vendor-specific switch platforms. A critical enabling technology for IN is Signaling System 7 (SS7), an internationally standardized channel signaling system for controlling switches and databases throughout the phone network. Service Switching Points (SSPs) intercept certain patterns of call processing steps to invoke service logic in Service Control Points (SCP). The service logic then influences the subsequent call processing steps. It is through such mechanisms that 800 number and call forwarding services are deployed in the Public Switch Telephone Network (PSTN). IN is intimately coupled to the hierarchical switching structure of the phone network and the logical sequencing of call processing which, in reality, are different among various switch vendors. Therefore it has failed to provide

service inter-operability across switches of different vendors. In addition, there is no elegant integration between fixed and mobile telephony services, let alone integration with other types of networks. Service creation in IN has a high cost-of-entry and is limited to a relatively small number of network operators [12] (more specifically, telecommunications service providers and *not* end users). In today's Internet environment, the IN approach is too closed, inflexible, complex, and lacks support for heterogeneity.

A similar approach is the ITU's H.323 "umbrella" standard [2]. The standards address the multimedia communications among terminals in packet switched networks. Beside a few built-in services, e.g., *call forwarding*, some supplementary services are defined, such as *Call Transfer*, *Hold*, *Park & Pick-up*, and *Call Completion on Busy*. The standard has no actual service programming interface and is quite limited in its IN-like approach of adding explicit service support inside its protocol.

In contrast with both IN and H.323, our model allows user level service creation and effectively supports heterogeneity.

B. Session Initiation Protocol

The IETF's Session Initiation Protocol (SIP) [3], is a lightweight protocol used for distributed communication session setup. Depending on the trust relationship between the service designer and the underlying communication system, two different approaches for service programming are under discussion. The first approach is analogous to the HTTP Common Gateway Interface and allows trusted users to execute scripts upon receiving SIP messages [4]. Untrusted users may customize their communication by using a specialized language, Call Processing Language (CPL) [5]. Similar to our approach, the language defines a set of basic primitives (such as "forwarding") and conditions (such as "on-busy" or time-dependent conditions), with which users can specify desired services.

Despite the improved openness and flexibility in SIP service creation, it is not obvious where the service logic resides and when it is executed. Intermediaries as well as endpoints may alter the communication process. This opens various possibilities for misconfiguration and undesirable interactions. In addition, it has not provided a well-defined system API and primitives, which complicates service portability.

In our system, we have a clean separation between user policies (user preferences, service configuration) and well-defined, extensible underlying system mechanisms (system APIs and primitives). Unlike SIP, our service logic does not run on any endpoints, but within the infrastructure. We believe this simplifies support for device and network heterogeneity as well as service reliability.

C. Other Approaches

JTAPI, developed at Sun Microsystems, offers a set of APIs for application programmers to access basic telephone call functionality from heterogeneous networks [8]. It concentrates on portable end application in contrast to portable communication services. It is not intended for customizing and extending communication services.

The Parlay consortium [13] offers a well defined call model, but is intended mainly for developers of enterprise telephony solutions (in contrast, we aim for end users). Parlay has a strong

focus on third party service provision and technology integration by using CORBA-like middleware which are not desirable for real time applications.

V. CONCLUSIONS

Internet-based unified communication systems are in need of a service creation model. In this paper, we presented such a model and framework. Central to our solution is the use of a network independent state machine for session establishment and control as well as exposing system parameters and primitives for user preference specification. By making multi-endpoint communications as the basic service of the communication system, it enables novel ways of new service creation.

Our state machines are simple because they are network independent. The simplicity in the state machines eases its standardization process across different vendors, and therefore simplifies service portability (namely, same service preference can be carried out in any systems that uses our service model). Encapsulating application-specific service agents (such as the billing agent) is a powerful third party service creation mechanism which is simple and flexible. By exposing appropriate system parameters and primitives, we allow arbitrary preference specification including inserting custom state machines into the basic model. As a first step evaluation of our model, we worked through most of the traditional telephony services. We also explored novel implementation of services like Pay-Per-View. These initial results look very promising.

REFERENCES

- [1] ITU-T Rec. Q 1200, *Q-Series Intelligent Network Recommendation Structure*, Geneva, Switzerland, September 1997
- [2] ITU-T Recommendation H.323 (Version 3), *Packet-Based Multimedia Communications Systems*, International Telecommunication Union - T Recommendations, Geneva, Switzerland, September 1999
- [3] M. Handley, H. Schulzrinne, E. M. Schooler, and J. Rosenberg, *SIP: Session Initiation Protocol*, Internet Engineering Task Force, Request for Comments 2543, March 1999
- [4] J. Rosenberg, H. Schulzrinne, and J. Lennox, *Common Gateway Interface for SIP*, Internet Engineering Task Force, Internet Draft, June 2000
- [5] H. Schulzrinne, and J. Lennox, *Call Processing Language Framework and Requirements*, Internet Engineering Task Force, Request for Comments 2824, May 2000
- [6] Anthony D. Joseph and B. R. Badrinath and Randy H. Katz *The Case for Services over Cascaded Networks* The First ACM/IEEE International Conference on Wireless and Mobile Multimedia, 1998
- [7] S. Petrack and L. Conroy, *The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services*, Internet Engineering Task Force, Request for Comments 2848, June 2000
- [8] Sun Microsystems, *Java (tm) Telephony Specification*, <http://java.sun.com/products/jtapi>, June 1999
- [9] Zhuoqing Mao, Helen J. Wang, Randy H. Katz, *Fault-Tolerant, Scalable, Wide-Area Autonomous Internet Service Composition*, submitted to publication
- [10] H. J. Wang, B. Raman et al., *ICEBERG: An Internet-core Network Architecture for Integrated Communications*, IEEE Personal Communications, Special Issue on IP-based Mobile Telecommunication Networks, Aug 2000
- [11] D.O. Keck and P.J. Kuehn, *The Feature and Service Interaction Problem in Telecommunications Systems: A Survey*, IEEE Transactions on Software Engineering, Vol. 24, No. 10, pp. 779-796, 1998
- [12] Jean-Pierre Hubaux et al *The impact of the Internet on telecommunication architectures*, Computer Networks: the International Journal of Computer and Telecommunication Networking, 1999
- [13] *The Parlay Group* <http://www.parlay.org/>