Accessing telephony services from the Internet

Vijay K. Gurbani and Xian-He Sun
Department of Computer Science
Illinois Institute of Technology
Chicago, Illinois
vkg@iit.edu, sun@cs.iit.edu

Abstract— **Networks exist to provide services to users. Increasingly, the networks on which the services reside are not the same as the networks from which the services are accessed. This leads to the problem on how to best provide such services transparently when the access protocols differ. We discusses a methodology to make available the existing services residing in a network whose protocols are distinct from the network where the access attempt is made. We propose a technique we term *call model mapping with state sharing* and demonstrate it in the telecommunication domain where we access traditional telephony services residing on the telephone network from Internet telephony endpoints residing on the Internet.**

Keywords- SIP; H.323; services; PSTN; call models; SS7; state machines

I. INTRODUCTION

The intrinsic value of a computer network is measured by the services it provides to its users. As the number of such networks increases, so do the chances that services residing on one network will need to be accessed by users on a different network. This is more true for telecommunication networks; currently, there are two major networks in place: the Public Switched Telephone Network (PSTN) and the Internet. Increasingly, these networks are converging [1], which necessitates access to services[1] residing in one of these networks from the other.

The convergence poses a number of problems to be solved. In order to formulate the problem set consistently, we will define some terms first. We call the network on which the service runs natively as the *local* network (or domain) ; alternatively, a *foreign* network (or domain) is one from which a request to execute the service is made. The service and its associated data reside on the local network.

The first problem in accessing services from a foreign domain is that of differing network protocols. The PSTN and Internet use dissimilar network protocols, and in fact, are designed with different goals. While the PSTN is a highly tuned network to transport voice, the Internet is a generalized network that can transport any type of payload -- voice, video, or data (text). Second, a request for service arriving from a foreign domain will start service execution in the local domain; as such, the entities in the local and foreign domain need to be synchronized. Third, when a service is accessed from a foreign domain, the semantics of the service must be preserved (i.e. the foreign network may have more capabilities -- or less capabilities -- than the local network; in either case, the service should function at a minimal acceptable level). Finally, local networks that host services have already addressed important issues such as scalability and reliability. There is a temptation to *port* services to the foreign network and then revisit the same issues in context of the foreign network. Instead, we believe that the services and their associated data and procedures are best left in the local network, with some technique created to allow access to these services in a transparent manner from the foreign network.

We describe a technique to address the problems outlined above. The technique is most applicable to domains where entities requesting and executing a service follow a finite state machine of some sorts. Call controllers -- entities that are responsible for setting up, maintaining, and tearing down a voice call, or a multi-media session -- in PSTN and Internet telephony readily subscribe to finite state machines. Thus, the telecommunications domain provides us a rich palette on which to focus the work described in this paper.

The rest of the paper is organized as follows: section II provides a background and motivation for our investigation in this area. Section III introduces our solution. The application of the technique to the telecommunication domain is presented in Section IV with emphasis on results obtained from an implementation. Section V outlines related work, and our conclusions are stated in the final section.

II. BACKGROUND

A. Motivation

The last few years have witnessed a rapid rise in academia and the industry in Internet telephony; i.e. using the Internet for real-time transfer of multi-media communications incorporating both traditional circuit switched and wireless networks and Internet applications such as the Web and email. Internet telephony itself is not new; it has been around since the 1970's. However, the pace of acceptance accelerated in the mid 1990's. As it stands right now, Internet telephony has moved beyond the hobbyist/recreational stage and is exceedingly being viewed as an additional means besides the PSTN for transporting voice.

[1] In the context of the PSTN and the Internet, we define *service* as a value added functionality provided to the users by the network; thus Call Waiting and Caller ID are examples of PSTN services and email and ftp are examples of Internet services.

The initial stages in the introduction of Internet Telephony (1995-2000) were characterized by a rush towards *toll bypass*; i.e. the price advantage gained by using unregulated Internet telephony over the highly regulated PSTN. As Internet Telephony matures, the thrust is moving from toll bypass to an important component: *services*. Services are an active area of research in Internet Telephony [11, 12, 13, 14, 22]. In this paper, we focus on services that reside on the PSTN but need to be accessed in a scalable and transparent manner from Internet endpoints.

In order to appreciate the need to access existing PSTN services from Internet endpoints, consider that the majority of services that PSTN users are accustomed to -- Call Waiting, 800-number translation, Caller ID, etc. reside on the PSTN. Users on an Internet telephony endpoint should be able to avail themselves of these services in the same transparent manner that they do when using a traditional PSTN handset. There are three ways [11] to accomplish this: first, the easiest, albeit the most intrusive way would be to re-write all the existing PSTN services for the Internet environment. This is not feasible since it takes anywhere from 6 months to a year to get a PSTN service specified, implemented, tested, and deployed. This already assumes a stable service delivery infrastructure as it exists in the PSTN. Internet telephony being a new medium does not as yet have a well-specified services architecture that can be leveraged to deploy new services. The service architecture for Internet telephony is in the early stages of being proposed [3,4,5,6]. Thus it is extremely difficult, and in fact, undesirable, to replicate existing PSTN services from scratch in the Internet domain.

Another way to enable services in dissimilar networks is to use a language neutral service framework. Much like a Java program can run on multiple architectures simply by porting the Java Virtual Machine to each such architecture, one can envision a service-specific language that is architecture and platform neutral. Thus services programmed in such a language can easily be ported from the PSTN domain to the Internet domain. Unfortunately, such a platform-neutral, domain-specific language does not exist. Although early research proved that such a language was indeed feasible [15], the industry interest was lacking to push it towards a standard.

A final option for accessing PSTN services in a transparent manner is to devise a technique such that services running on a local domain can transparently be accessed from foreign domains. This preserves the (tested and) deployed service infrastructure in the local domain, while at the same time, allowing transparent and scalable access to the service from the foreign domain. Service porting or re-writing is not necessary as the service can be accessed in a network agnostic manner.

B. Contribution

The main contribution of this paper is a technique we term call model mapping with state sharing (CMM/SS) to transparently access services resident in local domains from a foreign domain. This technique is most applicable when the services are accessed during the signaling messages exchanged at call setup, maintenance, and teardown.

The CMM/SS technique depends on, and assumes the availability of a call model. A call model is a deterministic finite state machine (FSM). States in the FSM represent how far the call has progressed at any point in time. The current state, plus a set of input stimuli transition the FSM to the next state. In telecommunication signaling, these input stimuli consist of timers firing and arrival/departure of signaling messages resulting in the execution of significant events. Events cause transition into and out of a particular state.

Fortunately, call models are already an intrinsic part of telecommunication signaling protocols. For instance, the PSTN call model consists of 19 states and 35 input stimuli [18,2,17] and the Internet telephony signaling protocol, SIP [7, 8, 9], consists of 8 states and 20 input stimuli (Figures 5 and 7 of reference [7]). Call models, besides providing a uniform view of the call to all involved entities, also serve to synchronize these entities.

CMM/SS consists of mapping the call model of a foreign domain to that of a local domain so that the foreign domain can access services resident in the local domain. Call mapping is fairly prevalent in the telecommunications domain [19,20,21], however, it has been used so far simply as an interworking function to set up calls between different networks (i.e. between PSTN and Internet, between SIP and H.323, etc.). Our work described in this paper extends this use of call mapping in two ways: first, the thrust is not on simply making a call across different networks, but *accessing services* across different networks. Secondly, consider that service execution in a heterogeneous network implies saving the state of the call in the foreign domain until the service has executed in the local domain. Thus, the state of the call needs to be shared between the foreign domain as well as the service execution function of the local domain. CMM/SS allows us to do this in a transparent manner.

We have successfully applied CMM/SS to access PSTN services from H.323 endpoints; reference [16] discusses that effort in detail. In this paper, we attempt to formally codify CMM/SS and, to demonstrate its generality, we apply it to another Internet telephony signaling protocol, SIP. The details of CMM/SS are described next.

III. CALL MODEL MAPPING WITH STATE SHARING

We begin by formally defining our state space and the call model mapping technique. We then take a look at how the state is effectively shared between the domains to access services.

A. Basic definitions

Definition 1 (localized state): A *localized state* is a finite tuple s^l of atomic states s_j, with $m \geq 1$, as shown in (1).

$$s^l = (s_j)_{j=1}^{m} = (s_1, s_2, \ldots, s_{m-1}, s_m) \quad (1)$$

Both foreign domains (*F*) and local domains (*L*) have their own localized states denoted by $F[s^l]$ and $L[s^l]$, respectively.

Definition 2 (global state): A *global state* is a finite tuple s^G of localized states $F[s^l]$ and $L[s^l]$ as shown in (2).

$$s^G = (F[s^l], L[s^l]) \quad (2)$$

The call model mapping technique intrinsically assumes that there are 2 domains of interest: the local domain, *L*, and the foreign domain, *F*. Furthermore, the states in $F[s^l]$ need to be mapped to the states in $L[s^l]$. We express the mapping process using the following notation:

$$(F \xrightarrow{j} L)_{j=1}^{m} \quad (3)$$

i.e. ∀ states in *F*, there exists a mapping between each state in *F* to an appropriate state in *L*.

If the notation in (3) is expressed as a function ∅(*x*), with *x* being the domain (or the set of argument values for which is ∅ is defined) of ∅ and $x \in F[s^l]$, then

$$\varnothing(x) = y, y \in L[s^l] \quad (4)$$

Or, $L[s^l]$ is the codomain of ∅(*x*). In other words, for every state in *F*, there must exist a possible (maybe non-unique) mapping in *L*. If any two random call models *F* and *L* exhibit (4) then they can be mapped to each other.

B. Call Model Mapping

In most mappings, the number of states in *F* and *L* may vary considerably. It could very well be that *L* has more states than *F*. In such cases, some states in *L* may not have an equivalent mapped state in *F*. We term this condition as state starvation and it is depicted in Figure 1.

On the other hand, it could very well be that *F* has more states than *L,* and thus, more than one state from *F* will map into a state in *L*. We term this condition as state aggregation and it is depicted in Figure 2. Regardless of state starvation or state aggregation, a mapping depicted in (4) is considered *complete* if there isn't any semantic loss of the protocol as a result of the call model mapping. A mapping is, likewise, considered *partially complete* if there is a minimal semantic loss introduced as the result of call model mapping due to state starvation or aggregation.

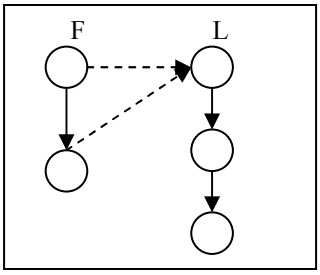

Figure 1: State Starvation

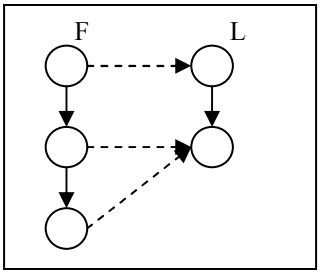

Figure 2: State Aggregation

It should be noted that unless two call models are exactly alike, all mappings will be partially complete. This can be attributed to the fact that the design of two randomly picked call models is rarely alike in every respect; thus mapping one into another necessarily introduces some semantic loss. However, so long as the mapping does not distort how the call model behaves in *L* if the mapping was not applied in the first place, partially complete mappings are indeed the only outcome of call model mapping.

Discussions of call model mapping in the literature thus far have not included any state sharing between *F* and *L*. State transition occurring in one domain cause the appropriate state transition in the other in an idempotent fashion. In fact, a majority of telecommunication systems that employ call model mapping [19,20,21] do so in this manner (for example, a PSTN 'User Alerting' message can be converted into a SIP '180 Ringing'). Call model mapping without state sharing is adequate for idempotent signaling messages that simply set up (or tear down) a session; no advanced services are involved during session setup (or tear down).

C. State Sharing

In order to access services resident in *L* using access protocols native to *L*, with the request for the service originating in *F*, we propose an addition to the call model mapping semantic to allow for state sharing across *F* and *L*.

Under our technique, state is actually distributed and shared between *F* and *L*. When a call request arrives at *F*, a state transition occurs to state $p \in F[s^l]$, and processing is temporarily suspended at *F* while the call state is handed to *L* for service execution (horizontal line from 'a' to '1' in Figure 3). This initial handoff is a simple mapping of $p \in F[s^l]$ into an equivalent state $p' \in L[s^l]$.

Since the services reside in *L*, they are executed in that domain. Their execution leads to state transitions local to *L* until a certain point that control needs to be passed back to the same state *p* from which the transition occurred (Figure 3). Along with passing the control, *L* also imparts enough information to *F*, allowing it to transition to a certain state $q \in F[s^l]$ (p != q and q may not be the next adjacent state after p). The choice of state *q* depends on the service execution logic. For instance, if the service logic in *L* decides to terminate the call, *L* will affect the appropriate state transition in *F*. Thus, in a sense, both *F* and *L* maintain global state s^G of the call. This synchronization is important because the call is actually being serviced by two different signaling protocols. The global state s^G reflects the shared and authoritative state of the call.

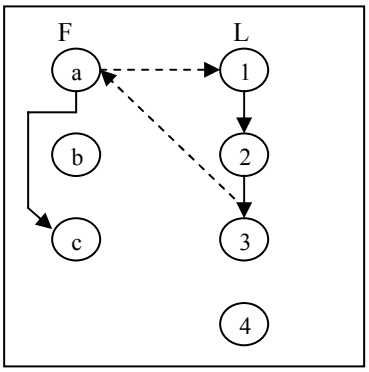

Figure 3: Transitions in F and L

Note that in Figure 3, the state labeled 'b' in *F* appears not to be mapped to an equivalent state in *L*. This is not an error, but in this particular example signifies that the service logic in *L* caused a state transition to occur to a non-adjacent state labeled 'c' in *F*.

We believe that the CMM/SS technique is a good fit for transparent access to services resident in *L*, using protocols native to *L*, when the service request originates in *F*. The call model and state sharing mechanism ensures that the different signaling protocols remain synchronized while the service is accessed and executed in *L*.

IV. IMPLEMENTATION

To prove the CMM/SS technique, we implemented it in a network consisting of SIP endpoints, a SIP proxy, and the PSTN service infrastructure. This section describes the implementation, with a quick detour on PSTN service execution for readers who may not be familiar with it.

A. Background: PSTN service execution

The Intelligent Network (IN) is used in the PSTN to provide services such as 800-number translation, pre-paid calling, etc. It is described in detail in [2]; here we provide a very rudimentary overview to aid the reader who may not be familiar with the concepts.

Figure 4 shows a simplified PSTN/IN architecture in which the telephone switches called Service Switching Points (SSP) are connected via a packet network called Signaling System 7 (SS7) to a general purpose computer called a Service Control Points (SCP). This leads to a clean separation of components since call control and service switching functions are performed at the SSP, while the service control (and data) functions are hosted by the SCP. Service logic resides at and is executed in the SCPs.

SSPs run a call model called Basic Call State Model (BCSM) when handling a call. A BCSM consist of states called Points In Call (PIC) and transitions between states called Detection Points (DPs). The DPs are armed to provide services; the SSP may suspend the call on encountering an armed DP and send a message to a SCP for further instructions. When the SCP gets a request for service, it can reply with a single response (a simple number translation augmented by criteria like time of day or day of week), or, in turn, get into a complex dialog with the SSP which may involve playing or recording voice announcements and collecting digits. The resulting protocol as well as the BCSM and DPs are standardized by the ITU-T and are known as the Intelligent Network Application Protocol (INAP) [2]. There are actually two BCSM: an originating side and a terminating side BCSM. Each BCSM provides services to a caller or a callee, respectively.

B. Implementation details

The goal of our research was to access from SIP endpoints, in a transparent manner, certain benchmark IN services which reside in the PSTN. The service logic and the protocol required to access it in the PSTN were assumed immutable; neither the logic, nor the protocol could not be modified or changed to account for the fact that the service request was now being sent by a SIP entity, and not a SSP.

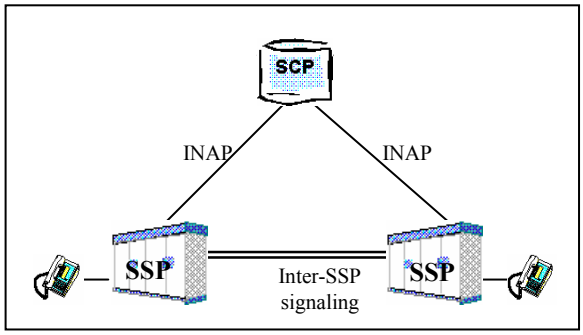

Figure 4: Simplified PSTN/IN architecture

The benchmarks IN services chosen to provide a proof of implementation were:

- Originating call screening (OCS): This service restricts the numbers that a phone subscriber can call. It can be used for parental control, or enforcing business practices. This is a caller oriented service (i.e. it is executed on behalf of the caller).
- Abbreviated dialing (AD): AD is one of the Virtual Private Network (VPN) service in the PSTN. It allows a caller to substitute a unique, abbreviated dial string to reach a party. This is a caller oriented service.
- Caller name delivery (CNAM): This service allows the called party to view the name of the caller in a display device. This is a callee oriented service.
- Call forwarding (CF): This service allows a phone subscriber to forward calls arriving at his/her phone number to an alternate destination. This is a callee oriented service.

Implementing a set of these 4 services, which are a mix between caller- and callee-oriented services provides an empirical proof for the technique of CMM/SS. The service logic programs for these services were executed on an SCP simulator. It is imperative to state that the service logic was not modified in any way; it was the same logic that would normally be used on the PSTN network. PSTN callers and PSTN callees were replaced by SIP-based callers and SIP-based callees.

The simplest manner in which to implement CMM/SS was to transform a SIP call processing entity into an Internet equivalent of a SSP. The best suited SIP entity for this purpose is a SIP proxy. The main task of a SIP proxy server is to route requests from one SIP endpoint to another based on many factors, including local registration information, DNS, SIP CGI, and SIP CPL [10]. Since a SIP proxy is already a central figure in the SIP network and is used by the endpoints to route session requests, it appeared logical to fortify it as an Internet SSP.

The problem, of course, is that SIP proxies do not run the PSTN/IN call models; they use a SIP protocol state machine. A SIP protocol state machine, depicted in Figure 5, is fairly simple: it can be modeled with 6 states and 7 transitions between them[2]. A Q.1204 PSTN/IN call model [2], on the other hand consists of 19 states and 35 transitions (11 states and 21 transitions in the originating BCSM, and 8 states and 14 transitions in the terminating one). Thus, a foremost task was to map the PSTN/IN call model into the SIP protocol state machine using the CMM/SS technique. A detailed discussion, including state-by-state analysis on how the mappings were established is documented in [17]. With respect to the discussion in section III, the mappings in [17] exhibit state starvation and are partially complete.

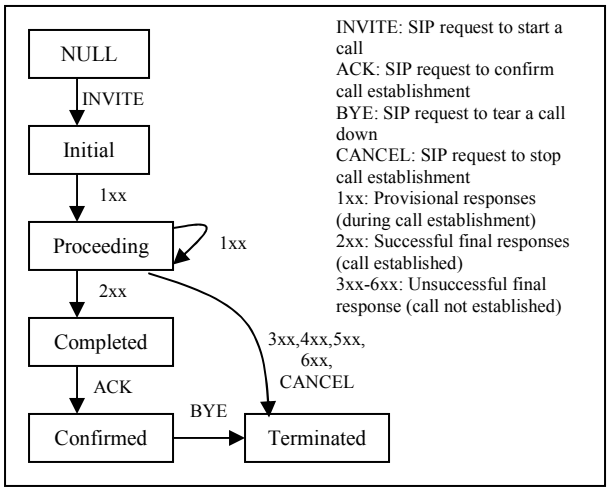

Figure 5: Modeling the SIP protocol state machine

We authored two pieces of software: first was a portable PSTN/IN call model which is a software layer written in C++, and the second was a proxy SIP server that had hooks to this call layer. After the mapping between the PSTN/IN call model states and SIP had been rigorously specified, the PSTN/IN call layer was then embedded into the SIP proxy.

In a SIP network with access to IN services, call requests are received by the SIP proxy, which intimates the portable IN call layer of this event through a functional interface, passing it parameters that include the caller's telephone number, the callee's telephone number and other pertinent information. The PSTN/IN call layer, acting as a SSP, steps through the PICs, and at appropriate points, executes the services by consulting a SCP to for further instructions. Services are executed by the PSTN/IN call layer sending an INAP request and transmitting it (over IP) to the SCP. The SCP responds with the pertinent answer encoded within the INAP response.

Once the call request has been thus serviced, control is returned back to the SIP proxy, which continues processing the call. The PSTN/IN call layer and the SIP proxy have to execute in lockstep since events can occur in either of the state machines to affect the other. For example, if the caller hangs up, the SIP proxy will get notified of this event first. It must now propagate this event to the PSTN/IN call layer so that it (the PSTN/IN call layer) can clean up any state associated with that call. Likewise, if the PSTN/IN call layer gets the notification to drop the call as a result of a service execution, it must notify the SIP proxy so it can clean state associated with the call.

Thus, an incoming call request is actually serviced by distributing its state across different call controllers, each of which used a dissimilar protocol. Call acceptance and final disposition was performed by the SIP proxy using the SIP protocol, while the services were accessed using PSTN protocols and PSTN call models.

C. *Network Topology*

Our laboratory setup consisted of several SIP endpoints (each running a SIP user agent client and a SIP user agent server), a SIP proxy server fortified with the PSTN/IN call layer, and an SCP simulation engine which serviced requests. Figure 6 depicts this setup.

Each SIP UA was configured, upon boot up, to register with the SIP proxy server using a telephone number. This is important; the intent is to mimic PSTN/IN services offered on the PSTN, hence endpoints are identified using telephone numbers and not the more powerful and generic email-like SIP URI. The SCP was configured with the data and logic pertaining to the benchmark IN services.

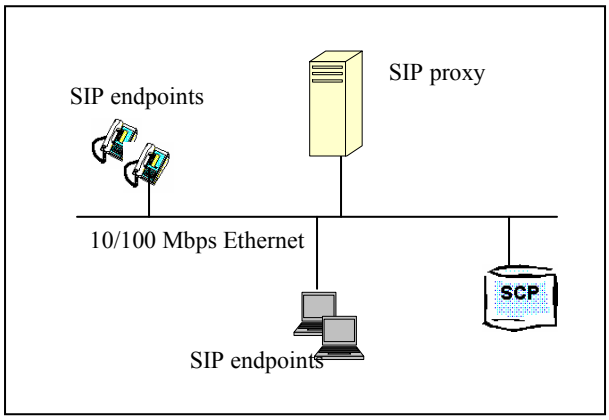

Figure 6: Network topology

D. *Service Details: OCS*

We now present a detailed overview of realizing a service based on the CMM/SS mapping technique. OCS is an originating side service wherein the PSTN/IN service logic ensures that the caller is authorized to place a call to the desired number.

In the PSTN/IN, this service is accessed by arming DP 5 (known as the Collected_Info trigger) of the originating call model. A partial mapping of the SIP protocol state machine and the Q.1204 PSTN/IN call model to realize this service is shown in Figure 7 (for a complete mapping, please see [17]).

[2] The 'Confirmed' state is entered through an ACK for a 2xx-class response since in this case the ACK is considered a separate transaction in SIP. For other responses, the ACK is part of the INVITE transaction and we have chosen not to explicitly model it with a state.

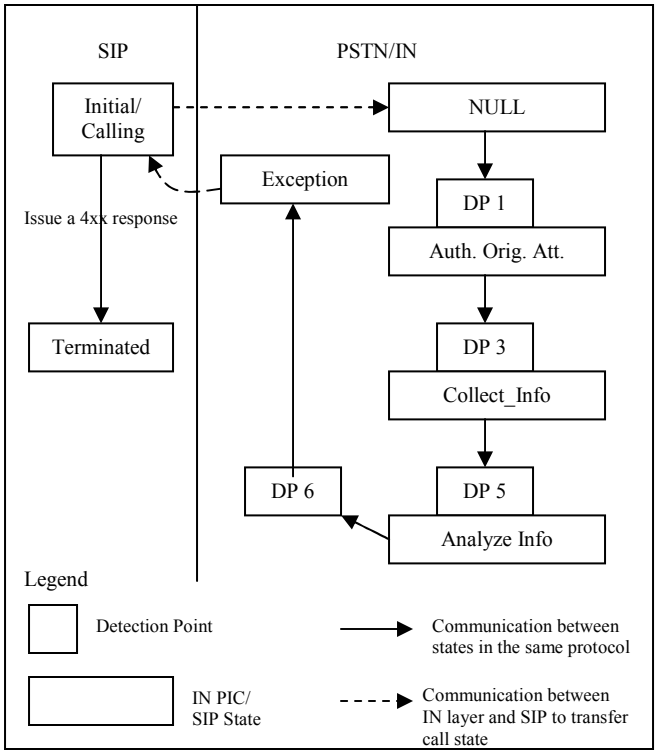

Figure 7: Partial mapping of SIP and PSTN/IN states

Given that the SIP protocol state machine has far fewer states than its PSTN/IN counterpart, the Initial/Calling state actually maps to more than one peer state in the PSTN/IN call model. Specifically, for the OCS service, the first four PSTN/IN states are mapped to the initial SIP state (the complete mapping in [17] actually depicts the initial SIP state mapping into the first seven PSTN/IN states).

The SIP proxy receives the incoming INVITE request and extracts the relevant headers from the request (the most important one for OCS are the Request-URI and the From header field; the Request-URI in SIP is used to specify the destination of the call and the From header is used to store the caller's information). Since the PSTN/IN services expect phone numbers, both the Request-URI and the From headers contain phone numbers. Once these headers are extracted, the SIP proxy executes the initial state transfer to send these to the portable IN call layer. Global state S^G is now distributed between the two domains with respect to the service execution. The services subscribed to by the caller are loaded by the IN call layer and processing continues as described next.

The first non-NULL state encountered in the IN call layer is the Authorize_Origination_Attempt state. In this state, the PSTN/IN confirms that the caller (From header) is indeed authorized to make a call. If the caller has the authority, control passes to the Collect_Info state through DP 3. In this state, the PSTN/IN is responsible for collecting a dial string from the caller and verifying the format of the string. In SIP, en-bloc signaling is used; i.e. the proxy gets the entire dial string in the Request-URI. Native PSTN endpoints, by contrast, signal the switch for every digit pressed. Thus, the Collect_Info state may be used to verify dial string before transitioning to Analyze_Info through DP 5. Since DP5 is armed, the portable call layer constructs an INAP request and transmits it to the SCP. Key fields in the INAP request contains information the portable IN call layer was passed through the initial state transfer from the SIP proxy. The SCP runs the OCS service and issues a response. If the caller identified in the From SIP header is not allowed to place a call to the number identified in the Request-URI, the call layer is so informed through the INAP response. Global state S^G is now updated as the IN call layer now passes control back to the SIP protocol state machine. The SIP protocol state machine examines S^G and issues an appropriate 4xx response.

E. Results

The results obtained from the implementation validate the CMM/SS technique. We summarize them in Table 1. The first column contains provisioning information required for the column contains provisioning information required for the service to operate (the 'data'). The second column contains the behavior of the service in the PSTN; and the third column details the behavior of the service in with the application of CMM/SS. The behavior in the third column is almost identical to that of the second column. This provides an empirical proof of validation of the CMM/SS technique.

TABLE I. CMM/SS RESULTS

Service	**Behavior**		
	Configuration data	***Behavior in PSTN***	***Behavior with CMM/SS***
OCS	A 'blocked number' list	Caller hears 'fast busy' tone	Call request rejected with '403 Forbidden'; some SIP user agents played a 'fast busy' signal on receipt of a 403
AD	Telephone numbers	The abbreviated number is expanded and routed to its destination	The PSTN/IN call layer returns a translated URI to which the SIP proxy routes the call
CNAM	PSTN name database	Callee's name is displayed in a caller ID device	Callee's name is displayed in the SIP UA GUI
CF	Telephone numbers	Incoming call is forwarded to a new destination	The PSTN/IN call layer returns a translated URI to which the SIP proxy routes the call

V. RELATED WORK

There has been work done at the Swiss Federal Institute of Technology (EPFL) on *hybrid* services [22]. These are services that span many network technologies. However, the intent of hybrid services leans towards establishing sessions rather than accessing IN services. Furthermore, [22] approaches the issue from an API point of view, not the

signaling and call modeling approach outlined in this paper. Other existing research in this area also focuses on APIs [23, 24, 25, 26, 27]. We eschew APIs in favor of direct access to signaling headers and call models since these provide the most amount of information to the developers. Services are best executed when the service execution platform has unfettered access to the signaling information; APIs tend shield the programmer from the details of the signaling protocol.

VI. CONCLUSIONS

We have presented a technique to access services in dissimilar networks. The entity making the service request is in a foreign network, in relation to the network which hosts the service (the local network). Thus, the state of a call request with respect to service execution is actually distributed across the two networks. In any distributed system, entity synchronization becomes an important component for the correct and deterministic functioning of such a system. The technique we present in this paper, CMM/SS, serves to distribute state across the networks and to synchronize the attendant entities as well. The global state of a call is maintained as a composite of each of the individual states. Consistency is imposed by forcing state transitions between the local and foreign networks.

The end result is that services written for one network can be successfully and transparently accessed by entities in another network, possibly using protocols unknown to the network hosting the service.

REFERENCES:

[1] Faynberg, I., Gabuzda, L., Lu, H-L. "Converged Network and Services: Interworking IP and the PSTN," *John Wiley and Sons Publishers,* January, 2000.

[2] Faynberg, I., Gabuzda, L., Kaplan, M., and Shah N., "The Intelligent Network Standards: Their Application to Services," *McGraw Hill Series on Telecommunications*, 1997.

[3] Rosenberg, J., "Distributed Algorithms and Protocols for Scalable Internet Telephony," *Ph.D. Thesis*, Graduate School of Arts and Sciences, Columbia University, New York, New York, 2001.

[4] Rosenberg, J., Mataga, P., Schulzrinne, H., "An Application Server Component Architecture for SIP," *IETF Internet-Draft*, Expired September 2001.

[5] Dianda, J., Gurbani, V., and Jones, M. "SIP Services Architecture," *Bell Labs Technical Journal*, Volume 7, Number 1, July 2002, Wiley Periodicals, Inc. pp 3-23.

[6] Ubiquity Corporation, "SIP Service Architecture", *White Paper*, <www.ubiquity.net/pdf/SIP-Service-Arch-WP1_0.pdf>, May 2001.

[7] Rosenberg, J., Schulzrinne, H., Sparks, R., Peterson, J., Johnston, A., Camarillo, G., Handley, M., and Schooler, E. "SIP: The Session Initiation Protocol," *IETF RFC 3261*, <http://www.ietf.org/rfc/rfc3261.txt?number=3261>, 2002.

[8] Sinnreich, H., Johnston, A. "Internet Communications using SIP," *John Wiley & Son Publishers*, October 2001.

[9] Camarillo, G., Rosenberg, J., "SIP Demystified," *McGraw Hill Publishing Company*, August 2001.

[10] Gurbani, V., Chiang, T-C., Kumar, S., "SIP: A Routing Protocol," *Bell Labs Technical Journal*, Volume 6, Number 2, July-December 2001, Wiley Periodicals, Inc.

[11] Gurbani, V., "Enabling Services Through Protocol Interworking", *Ph.D. Proposal*, Department of Computer Science, Illinois Institute of Technology, Chicago, Illinois.

[12] Rosenberg, J., Lennox, J., and Schulzrinne, H., "Programming Internet Telephony Services", *IEEE Internet Computing,* Vol 3., No. 3, 1999, pp 63-72.

[13] Lennox, J., and Schulzrinne, H., "Feature Interaction in Internet Telephony", *Proceedings of Feature Interaction in Telecommunication and Software Systems VI*, Glasgow, UK, May 2000.

[14] Lennox, J., Schulrzrinne, H., and La Porta T., "Implementing Intelligent Network Services with the Session Initiation Protocol", Columbia University Technical Report No. CUCS-002-99, unpublished, accessed from <http://www.cs.columbia.edu/~lennox/cucs-002-99.pdf>

[15] Slutsman, L., Lu, H-L., Kaplan, M., and Faynberg, I., "The Application Oriented Parsing Language (AOPL) as a way to achieve platform-independent service creation environment," *Proceedings of the Third International Conference on Intelligence in Networks (ICIN94)*, October 11-13, 1994.

[16] Chiang, T-C., Douglas, J., Gurbani V., Montgomery, W., Opdyke, W., Reddy, J., and Vemuri, K. "IN Services for Converged (Internet) Telephony", *IEEE Communications Magazine*, Vol. 38, No. 6, June 2000, pp 108-115.

[17] Gurbani, V., Haerens, F., and Rastogi, V., "Interworking SIP and Intelligent Network (IN) Applications", *IETF Internet-Draft*, Expires December 2002, Work in Progress, <http://search.ietf.org/internet-drafts/draft-gurbani-sin-02.txt>

[18] ITU-T Q.1204: "Intelligent Network Distributed Functional Plane Architecture", *ITU-T Recommendation Q.1204*, 1993.

[19] Camarillo, G., Roach, A., Peterson, J., Ong, L., "ISUP to SIP Mapping", *IETF Internet-Draft*, Expires December 2002, Work in Progress, < http://www.ietf.org/internet-drafts/draft-ietf-sipping-isup-03.txt>.

[20] Singh, K., Schulzrinne, H., "Interworking between SIP/SDP and H.323", *Proceedings of the 1st IP Telephony (iptel) Workshop*, 2000.

[21] Vemuri, A., and Peterson, J., "SIP for Telephones (SIP-T): Context and Architectures", *IETF RFC 3372*, < http://www.ietf.org/rfc/rfc3372.txt>, September 2002.

[22] Gbaguidi, C., Hubaux, J.P., Pacifici, G., and Tantawi, A.N., "*Integration of Internet and Telecommunication Services: An Architecture for Hybrid Services*", IEEE Journal on Selected Areas in Communications, August 1999.

[23] Anjum, F. Caruso, R. Jain, P. Missier, and A. Jordan, "*ChaiTime: A System for Rapid Creation of Portable Next-Generation Telephony Services Using Third-Party Software Components*", Proceedings of the 2nd IEEE Conference on Open Architectures and Network Programming (OPENARCH), New York, USA, March 1999.

[24] Moyer, S., and Umar, A., "*The Impact of Network Convergence on Telecommunications Software*", IEEE Communication Magazine, Jan 2001, pp 78-84.

[25] Bergmark, D., and Keshav, S., "*Building Blocks for IP Telephony*", IEEE Communications Magazine, April 2000, pp 88-94.

[26] De Keijzer, J., "*Intelligent Agents and Java Advanced Intelligent Network Architecture (JAIN)*", Proceedings of IATA, 1998, <http://sokrates.cs.tu-berlin.de/deutsch/news/tagungen/IATA89/main.html>

[27] Low, C., "*Integrating Communication Services*", IEEE Communications Magazine, Volume 35, Number 6, June 1997.