ICEBERG: An Internet-core Network Architecture for Integrated Communications

Helen J. Wang, Bhaskaran Raman, Chen-nee Chuah, Rahul Biswas,
Ramakrishna Gummadi, Barbara Hohlt, Xia Hong, Emre Kiciman,
Zhuoqing Mao, Jimmy S. Shih, Lakshminarayanan Subramanian,
Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz
http://iceberg.cs.berkeley.edu/
Contact: helenjw@cs.berkeley.edu
Computer Science Division, U. C. Berkeley

Abstract— **In the ICEBERG project at U. C. Berkeley, we are developing an Internet-based integration of telephony and data services spanning diverse access networks. Our primary goals are extensibility, scalability, robustness and personalized communication. We leverage the Internet's low cost of entry for service creation, provision, deployment, and integration. In this article, we present our solutions to signaling, easy service creation, resource reservation, admission control, billing and security in the ICEBERG network architecture.**

Keywords— **integrated communication, service creation, scalability, availability, fault tolerance, customization, cluster computing platform, signaling, resource reservation, admission control, billing, security, privacy, authentication.**

I. INTRODUCTION

Telecommunications networks are migrating towards Internet technology, with voice over IP maturing rapidly. We believe that the key open challenge for the converged network of the near future is its support for diverse access technologies (such as the Public Switched Telephone Network, digital cellular networks, pager networks, and IP-based networks) and innovative applications seamlessly integrating data and voice. The ICEBERG Project at U. C. Berkeley is seeking to meet this challenge with an open and composable service architecture founded on Internet-based standards for flow routing and agent deployment. This enables simple redirection of flows combined with pipelined transformations. These building blocks make possible new applications, like the *Universal In-box*. Such an application intercepts flows in a range of formats, originating in different access networks (e.g., voice, fax, e-mail), and delivers them appropriately formatted for a particular end terminal (e.g., handset, fax machine, computer) based on the callee's preferences.

We designed ICEBERG, an Internet-core network architecture for integrated communications, to meet these goals:

- **Potentially Any Network Services (PANS):** This means that any service can be accessed *transparently* from any end-device via any access network, such as accessing e-mails via a cellular phone. To achieve this goal, we make our system design and implementation *network and device independent*, which allows new networks and their access devices to be plugged into the ICEBERG architecture without changes to the system.
- **Personal Mobility:** This is the concept of having the person (and not the communication device) as the communication endpoint[1]. By using a single identity for an individual, we can implement a level of indirection to the desired endpoint for communication. Personal mobility is a key motivation for integrating services across heterogeneous devices from diverse networks.
- **Service Mobility:** This refers to seamless mobility across different devices in the middle of a service session (for example, switching from a cell-phone to an IP-Phone in the middle of a conversation).
- **Easy Service Creation and Customization:** The design of the signaling protocol directly affects the ease of implementing call processing services, such as call forwarding, call waiting, etc. Easy service creation also requires the system's assistance for resource reservation, admission control and integrated billing for the new services. Service customization requires user preference management and user activity tracking.
- **Scalability, Availability and Fault Tolerance:** We aim to scale our system incrementally to support a large user base (i.e., large geographic regions and hundreds of thousands of simultaneous calls). The components of the network should be available 24 hours a day and 7 days a week. The architecture should be able to tolerate failures gracefully and hide them from end-users. To this end, we leverage Ninja [21], a companion project at Berkeley, which offers a cluster[2] computing platform for scalable, available, and fault tolerant services. We envision our network as islands of ICEBERG-capable Ninja cluster computing platforms which act as single large-scale computers, with a robust signaling protocol running between them.
- **Operation in the Wide-area:** The challenges in the operation of the ICEBERG network in the wide-area are managing network partitions and achieving quality-of-service (QoS). Our protocols must detect network partitions and react to them promptly. QoS directly relates to resource reservation, admission control and billing. We tackle these problems with a distributed *Clearing House* architecture that selects the best packet traversal routes across various Internet Service Providers (ISPs), makes aggregated resource reservation and maintains billing information for each user.
- **Security, Authentication and Privacy:** Security and privacy issues related to personal mobility are critical and require careful attention especially in ICEBERG, which follows an open service model in the Internet. We use a hybrid of shared and asymmetric key cryptography at different stages of call setup or generic service access. We apply optimizations at different levels to bring down network

[1] The concept originally comes from Personal Communication Services [40]. The Mobile People Architecture [9] also identifies this principle.

[2] A cluster refers to a number of PCs interconnected by high-speed network.

round-trip times as well as computational requirements.

For the rest of the paper, we first discuss the related work (Section II). Next, we introduce the ICEBERG architecture (Section III). Then we describe our signaling protocol which performs the call setup and control (Section IV). The key features of our signaling protocol are that it captures all call session dynamics (such as membership changes), tolerates component failures and transient network partitions and detects the prolonged network partitions. Our signaling protocol supports the multi-device communication as a first-class service. In Section V, we discuss how data path creation enables easy service creation, then we present how our signaling primitives simplify the implementation of services that require endpoint changes including a novel service called *service handoff* which allows users to switch devices in the middle of a call. Also, in the same section, we demonstrate how ICEBERG components simplified our tasks of providing the *Universal In-box service*, the Ninja Jukebox service and the Room Control Service. We illustrate our initial solutions to resource reservation, admission control, and billing through the use of the Clearing House architecture in Section VI. Then we describe our security measures in Section VII and present our implementation and performance evaluation in Section VIII. Finally we discuss our future work and conclude in Section IX.

II. Related work

A wide assortment of commercial products that provide service-integration are becoming available: e-mail-to-fax services [20] [24], voice-e-mail-fax integration services [22] [26], and enhanced telephony services [27]. These commercial services show the strong desirability of having personalized, integrated communication. While they address specific integrated services, the ICEBERG architecture provides building blocks and infrastructure for enabling any type of services.

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 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. Finally, service creation in IN has a high cost-of-entry and is limited to a relatively small number of network operators [13] (more specifically, telecommunications service providers and *not* end users).

The IETF PINT working group, in particular its WebIN architecture [31], identified the difficulty of service creation in IN. They proposed running the service logic in the Internet (in other words, SCPs are part of the Internet implemented by non-proprietary programming or scripting languages). Nonetheless, the difficulty of integrating with other non-telephony networks remains. We believe that this difficulty is inherent in PSTN-core based networks. ICEBERG takes an Internet-core based approach. By isolating the access network specific functionalities in only one component of the system, adding new networks is simplified in ICEBERG. In addition, ICEBERG eases the creation of services, including novel and sophisticated services that are beyond telephone call features, through the flexible composition of computation units, powerful signaling protocol primitives, and preference management and user activity tracking components (Section V).

Hybrid services, in [12], are defined as services that span different network technologies, similar to our “PANS” concept. The hybrid service architecture also takes an Internet-core based approach. Its network is composed of managed networks (“clouds”) interconnected by their edge gateways. Each cloud has a service platform, a middleware layer to provide a set of commonly needed functional components. A cloud is analogous to an ICEBERG-capable Ninja cluster computing platform. While their architecture is similar to ours at a high level, the detailed mechanisms of preference management, name mapping, location tracking, signaling, and novel services have not yet been discussed.

TOPS [3] is a packet telephony architecture which includes a directory service, application layer signaling protocol, a logical channel abstraction, and an encapsulation format to insulate telephony applications from the underlying network transport capabilities, and mechanisms to support a variety of conferencing modes. The directory service is the key component of the system. It manages users’ call receiving preference, performs name mapping between terminal addresses and unique names and tracks current user location. These three functionalities are separated into different components in ICEBERG (namely, Preference Registry, Name Mapping Service, and Personal Activity Coordinator) for better modularity and more effective data management since these three types of data have different dynamics. Conference control in TOPS is accomplished using an expanded set of its signaling protocol, while in ICEBERG, the basic signaling readily supports multi-party calls. Issues on scalability, availability, fault tolerance, and wide-area operations are not addressed in TOPS, which we do in both our architecture design and the signaling protocol design.

The Mobile People architecture (MPA) [9] [35] tackles the problem of personal mobility by introducing a *person layer* on top of the application layer emphasizing the idea that the person, rather than the device, is the communication endpoint. Each person is identified by a globally unique *Personal Online ID*. Each *Personal Proxy* is associated with a person and performs person-level routing including location tracking, accepting communications on a person’s behalf, converting the communications into different application formats according to preferences, and forwarding the resulting communication to the person. The use of a Personal Proxy achieves location privacy. While a Personal Proxy is independent of the existing network and telecommunication infrastructure and is easy to extend to new devices and networks, one drawback is that regardless of where a person is, all communication to the person must go through her Personal Proxy, which can yield inefficient routes. This can only be prevented by intercepting the call signaling and resolving the callee destination during the call setup instead of after the call setup to the Personal Proxy in the MPA. This capability requires support from the infrastructure. ICEBERG architecture provides such support by managing user preferences and performing location tracking in the core infrastructure rather than at endpoints. Here, we provide a trustworthy infrastructure with appropriate security mechanisms. In addition, we preserve location privacy by not revealing user locations outside their administrative domains, namely, their ICEBERG service provider (Section III).

For scalability, availability, and fault tolerance, we leverage cluster computing platforms like Ninja [21] and Active Service (AS1) [1]. Although each intends to be a general cluster computing environment that support any cluster-based services, Ninja targets long running services with infinite lifetimes (such as web servers), while AS1 focuses on services with finite session lifetimes (like video-conferencing). ICEBERG contains both kinds of services. We augment Ninja with AS1 features for session-based services (Section III-A).

In terms of the signaling protocols, the ITU's H.323 [23] and the IETF's Session Initial Protocol [37] are the dominant Internet Telephony signaling protocols. Schulzrinne, et al., in [36], provides an extensive evaluation of H.323. H.323 is complex due to its numerous component protocols, hard to extend due to its requirement of full backward compatibility between versions and centrally registered codecs, non-scalable due to its use of stateful TCP connections for signaling transport and a central control for conference calls, and limited in preference support for customizable services. Both SIP and our signaling protocol provide significant improvements on these aspects. Neither SIP nor H.323 have addressed fault tolerance and scalable mechanisms of tracking accurate membership in a multi-party call, which we do (Section IV).

Several bandwidth broker implementations have been proposed in [34] as a scalable mechanism for QoS provisioning over a Diff-Serv (RFC 2475) architecture. In [17], M. Günter et al. presented the broker signaling trade-offs in [8], but they do not optimize end-to-end path selections. The Internet2 QoS working group operates Qbone [25], an inter-domain Diff-Serv testbed. They have started to investigate the inter-broker signaling to automate the adaptive reservation scenario, but currently the brokers are configured manually. Duffield et al. describe in [10] an adaptive reservation scheme that is optimized for virtual private networks (VPNs), and compare its performance to static provisioning using real traffic traces. However their work only considers a single ISP scenario. Our Clearing House (CH) design (Section VI) provides a scalable approach for inter-ISP signaling and coordinates resource reservations between multiple ISPs dynamically. The CH uses a capability based security scheme, similar to Kerberos [38], that requires users to present tickets to use ISP services. We picked a capability based security scheme instead of an access control based scheme to ensure fast call setup time. The CH also provides secure billing services, which, to our knowledge, have not been addressed before.

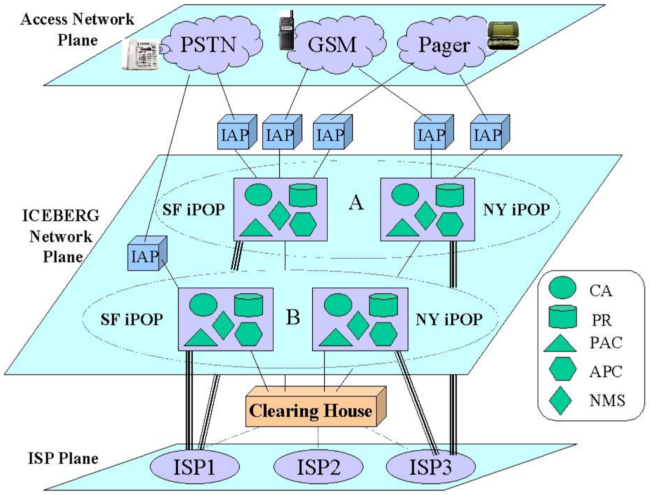

Fig. 1. ICEBERG Architecture Overview

III. ICEBERG Architecture

Figure 1 depicts the ICEBERG architecture. The ICEBERG network plane shows two *ICEBERG Service Providers* representing two different administrative domains: A and B. The services they provide are customizable integrated communication services on top of the Internet. Similar to the way that Internet Service Providers (ISP) provide Internet services through the use of Points of Presence (POPs) at different geographic locations[3], ICEBERG Service Providers consist of ICEBERG Points of Presence (iPOPs). Both A and B have iPOPs in San Francisco (SF) and New York (NY). Each iPOP contains Call Agents (CAs) which perform call setup and control, an Automatic Path Creation Service (APC) which establishes data flow between communication endpoints, a Preference Registry (PR) for users' call receiving preference management, a Personal Activity Coordinator (PAC) for user location or activity tracking, and a Name Mapping Service (NMS) which resolves user names in various networks. iPOPs must scale to a large population and a large call volume, be highly available and be resilient to failures. This leads us to build the iPOP on the Ninja cluster computing platform. The ICEBERG network is an overlay network of iPOPs on top of the Internet.

ICEBERG Access Points (IAPs) are the gateways to the access networks, such as the PSTN, GSM cellular networks, and the pager networks. They serve as bridges between the access network plane and the ICEBERG network plane.

Each iPOP employs an Internet Service Provider (ISP)[4] (for example, in Figure 1, SF iPOPs of both A and B employ ISP1, NY iPOPs of A and B employ ISP3). The ISP makes service level agreements and peering arrangements with other ISPs to carry the traffic among iPOPs. The Clearing House serves as a bandwidth broker and accountant for iPOPs and interacts with the ISPs.

Establishing a new ICEBERG Service Provider, is simply a

[3] ISPs generally locate POPs such that users can make a local call and gain Internet access.

[4] Please note that ICEBERG service providers are completely orthogonal to Internet Service Providers

matter of creating an iPOP with an NMS, PR, PAC, and APC service running on it, and IAPs for the access networks to be supported. Different service providers may share their components, such as the IAPs, based on some pre-established agreements or arrangements.

In our system, the NMS, PR, and PAC are used in the control path. The APC service establishes and manipulates data flows (for example, voice streams). The IAP deals with both control path (signaling translation) and data flow (voice stream packetization).

For the rest of the section, we first describe how we leverage the cluster computing platforms. Then we describe each ICEBERG component in detail. Finally, we present a scenario to illustrate how ICEBERG components interact with one another.

A. Leverage Cluster Computing Platforms

Each iPOP requires processing scalability to a large call volume, 24×7 availability[5] through fault masking, and cost-effectiveness. Clusters of commodity PCs interconnected by a high-speed System Area Network (SAN), acting as a single large-scale computer [2] (assuming no network partitions within a cluster), are especially well-suited to meeting these challenges [16]. Cluster computing platforms like the Ninja Base [11] and Active Service Platform (AS1) [1] provide an easy service development environment for service developers and mask them from cluster management problems of load-balancing, availability, and failure management. We describe briefly the underlying mechanism of both platforms and how we leverage them.

In a Ninja *Base*, each node houses an execution environment into which mobile code can be pushed. Services have many instances within the cluster for fault tolerance, but clients are shielded from this by using a service *Redirector Stub* to interact with the cluster. The Redirector Stub for a service is dynamically generated at run-time and contains the embedded logic for clients to select one service instance from a set of nodes within the cluster. Within the Base, all nodes maintain a replicated registry of all service instances; each node periodically sends out a multicast beacon carrying its list of local services to all other nodes in the Base and listens for the multicast beacons from other nodes. The Base also maintains persistent service states. As a result, the failure of a service instance on one node automatically triggers another service instance to take over for the failed one. To the outside world, a Base acts like a non-distributed, robust, highly-available, and high-performance service.

AS1 platform differs from the Ninja Base approach in that service clients send keep-alive service requests instead of using redirector stubs. AS1 maintains a table mapping each service request to its associated service agent. A new service request causes AS1 to spawn a new service agent, and to enter the service request and agent pair into the table. A timeout on a service request in the table indicates the end of the service lifetime, and causes the table entry to be removed. A failed service agent will also cause its entry to be removed from the table just the same as in a Ninja Base. Service requests contain the current session state, and service agents are entirely soft state.

We run iPOPs on Ninja Bases which should have at least two nodes in the cluster for load-balancing and fault tolerance. In addition, we have added support for the keep-alive service request from AS1: call requests are the service requests; Call Agents are service agents. We run the Preference Registry, Name Mapping Service, Personal Activity Coordinator, and Automatic Path Creation Service on Ninja Bases, which handle the fault detection and recovery of these components. For the rest of the paper, "iPOP" refers to an ICEBERG-capable Ninja Base.

B. Name Mapping Service

Personal Mobility is a key mechanism for managing communication across heterogeneous devices from diverse networks. The idea of *Personal Mobility* is to make the person[6], and not the communication device, be the communication endpoint [9] [40] The concept is becoming more important as people acquire more and more end devices. By using a single identity for an individual, ICEBERG provides a level of indirection to the desired communication endpoint.

In ICEBERG, we associate each user with an ICEBERG Unique ID (iUID). The structure of iUIDs is still an open problem in ICEBERG. For now, an iUID takes the form of an e-mail address. The *Name Mapping Service* maintains the mapping between the users' various communication endpoints and their iUIDs.

C. ICEBERG Access Point

An ICEBERG Access Point (IAP) is a gateway that interconnects an access network with the rest of the ICEBERG network. It consists of hardware and software for transcoding between the signaling protocols and the data formats used by the access network and those used by the ICEBERG network. For keeping track of each stage of the call initiation process, an IAP maintains call state machines for all the calls (either inbound or outbound) handled by it, in cooperation with the access network signaling protocol (like SS7); and the IAP issues ICEBERG call requests to the iPOPs.

Running an IAP on a Ninja Base would provide minimal benefits, as an IAP includes a network-specific gateway that may contain hard state for each call (a function of the signaling protocol used by the access networks). Such access network-specific state cannot be replicated, thus the failure of an IAP causes all calls handled by it to be dropped (as a comparison, the failure of a Base Station Controller in the GSM network would cause all calls handled by it to be dropped). Thus, it is the gateway's responsibility to provide the appropriate level of fault tolerance for its access network.

D. Call Agent

A *Call Agent* is a service agent that performs call setup and control for a caller's device. A Call Agent interacts with the Clearing House for resource reservation, call admission control and accounting, with a Name Mapping Service to resolve caller or callee identities or addresses, with the Preference Registry for the desirable call receiving device, and with the Automatic Path Creation Service for data flow establishment. There is one Call Agent per device per caller. An iPOP spawns a Call Agent

[5] 24×7: 24 hours a day and 7 days a week.

[6] It is possible to name roles (e.g., department chair, graduate admission) as well as individuals.

on a node in the cluster whenever a call request (for either an inbound or outbound call) arrives at the iPOP. The Call Agent is kept alive by the periodic call requests from its client (the device or its IAP). The Call Agent is terminated when the call requests cease after the device hangs up.

We choose not to build Call Agents or iPOPs as part of the IAPs for three reasons. First, call setup and control within the ICEBERG network is the same for all access networks. It is not desirable to duplicate this functionality in the IAPs of each access network from the software engineering point of view. Second, since IAP failures cause call drops, we want to minimize the amount of logic in IAPs, and therefore make them less failure-prone. Lastly, building iPOP into an IAP does not scale to a large number of calls as an access network gateway contained in an IAP can only handle a limited number of calls, while an iPOP as a separate entity can handle calls from many IAPs. The benefit of this decoupling is that IAPs are the only access network dependent components. All other ICEBERG components are designed in a *network and device independent* fashion. This greatly simplifies the expansion of ICEBERG to new networks and devices.

E. Automatic Path Creation Service

The Automatic Path Creation Service (APC) establishes data flows between communication endpoints. A *data path* is an abstraction of a data flow – a set of transcoding *operators* strung together using *connectors* [21]. An operator is a unit of computation on some data, and a connector is an abstraction of the Application Data Unit (ADU) transport mechanism between two operators (e.g., RTP connector, UDP connector, etc.). For instance, a series of codec operators followed by a speech-to-text operator is a path. *Data path* is a powerful concept for enabling communication among heterogeneous end-devices because of its flexible service composability.

In our architecture, Call Agents make requests to the APC service for data path creation. The Call Agents provide the APC service with endpoint information, such as the endpoint's address (e.g., IP address and port number for a desktop phone, the phone number and cellular gateway address for a cell phone) and transport protocol (e.g., RTP). The employment of APC encapsulates the data path creation and instantiation process from the rest of the system, and cleanly separates data from control.

F. Clearing House

The Clearing House (CH) is a third-party entity (outside the ICEBERG Service Providers) that manages the traffic flow between the ICEBERG Network plane and the ISP Plane (Figure 1). This entity interprets traffic specifications received from the iPOP, and searches for the optimal packet flow path from the sender to the receiver on the ISP plane. The CH also maintains resource reservations across ISPs, performs admission control and provides authentication and secure billing services to the ICEBERG architecture. We assume that the CH can be trusted by both the ISPs and ICEBERG, and ISPs cooperate with the CH for resource reservation. We illustrate the CH architecture in more detail in Section VI.

Note that the packet flow path is different from the data path described in the previous section: the former describes the sequence of ISPs the packets traverse, while the latter refers to a sequence of operators associated with connectors. We will show in Section V-A that each data path is contained in one iPOP, so the operators and connectors of a path live on the same iPOP.

G. Preference Registry

To achieve the goal of customizable communication services, we need mechanisms for user preference management. The *Preference Registry* is such a service that stores and processes user preferences. It takes as input the caller ID, callee ID, time of day, and dynamic user information (e.g., current location or current user behavior), and returns the callee's preferred endpoint (e.g., cell-phone number or e-mail address). During call setup, the Preference Registry is queried for a callee's preferred endpoint. Users convey their preferences (such as when they want to be called, on what device, by whom) using a tool that generates scripts (e.g., Perl, Tcl, or some call processing specific scripting language) and stores them in the Preference Registry.

H. Personal Activity Coordinator

The *Personal Activity Coordinator* (PAC) assists the Preference Registry for more powerful service customization. The PAC tracks a user's current activity (for example, "Alice is on her cell phone with Bob, who is a VIP") and other dynamic events, such as a user's current location, or the call status of a device – busy, on-hold, etc. The PAC allows users to control privacy policies, such as which information is tracked and to whom the information can be released. The information maintained in the PAC is used by the Preference Registry as additional inputs and hints in processing user preferences. This information is stored as soft state. Some default user behavior is assumed if the PAC does not provide any information.

I. An Illustration

In this section, we illustrate how ICEBERG components work together to provide a personalized call handling service through a scenario: Alice uses her cell phone to dial Bob's cell phone number. Bob currently prefers to receive phone calls from Alice on his office PSTN phone.

The call from the cell phone is intercepted at an IAP (or one could imagine that Alice first dials into an IAP). The IAP locates[7] an iPOP for Alice and issues an ICEBERG *call request* to the iPOP. The iPOP informs the CH about the call request so that the Clearing House can make resource reservations, perform admission control and do accounting. If the call is admitted, the iPOP spawns a CA to serve Alice's cell phone. The CA first finds the identity of the called party (i.e., Bob), as well as the location of Bob's Preference Registry, using the Name Mapping service. It finds Bob's preferred endpoint, and returns an appropriate iPOP for reaching the callee[8], from his Preference Registry. The callee iPOP also interacts with the CH for resource reservation, admission control and accounting. Next, Bob's iPOP spawns a CA to serve Bob. This CA then locates a PSTN IAP to ring Bob's office phone. When Bob picks up the phone, the CAs interact with the APC service to establish the data path. From this point on, Alice and Bob are in conversation.

[7] This can be done through some service discovery service.

[8] Please note that the preferred call receiving endpoint itself is not returned for user privacy.

IV. The ICEBERG Signaling System

A signaling system is a collection of network elements, Call Agents (CA) in our case, that use a communication protocol to effect call setup, routing, and control. In this section, we present the ICEBERG signaling system. We follow the same approach of separating signaling from data as in many other communication systems. Also, our signaling protocol is independent of the access networks. Detailed security measures for signaling are presented in Section VII.

Current telecommunication systems designed their signaling protocols, such as SS7 for the PSTN, to support two-party calls with homogeneous devices (i.e., telephones), as the *basic call service*. Additional services, such as call forwarding and conference calls, are implemented through overriding, augmenting, and reusing the basic call signaling primitives. As a result, these services incur extra expenses to users. In ICEBERG, we aim to provide more sophisticated basic services with little cost to users while enabling more powerful service building primitives. Our architecture supports multi-device communication as the basic call service and allows it to become a common case. A multi-device call can involve an unlimited number of participants and heterogeneous devices[9] (for example, using an audio tool and a video tool together for a call). These primitives make services, such as conference calls, call forwarding, and a novel service called service handoff, trivial to implement. We detail this in Section V. We assume that in a multi-device call, the call participation is invitation-based, and not subscription-based. Any call participant can invite new parties to join the call.

A multi-device call can be highly dynamic. New devices (or new call parties) may be invited to join the call (possibly simultaneously), and they must establish data flows with each call participant. A participating device may leave the call, and its CA needs to tear down a portion of the data path. In addition, CAs that serve the call may fail, then get restarted with a new identity at a new location. The new CA must obtain the current state of the call (such as the current call membership and device status) including the state changes that occurred during the fault recovery process. Network partitions may also occur, and then are healed. The state changes occurring in each partition must be captured by the participants. The signaling protocol must address the issues of maintaining accurate call membership and capturing the complete call state in face of any faults and in a scalable fashion. To our knowledge, these have not been addressed effectively in other Internet Telephony signaling protocols.

For call membership, we introduce the notion of a *call session* among a collection of communicating CAs. The communication is broadcast messages to a shared group communication channel per call session, rather than pairwise communications between CA pairs. This yields scalability to a large number of call participants. The channel offers a level of indirection that hides the identity and the location of the CAs and relieves each CA from maintaining the dynamic session membership – this effectively addresses the issue of membership dynamics.

Now we turn to the issue of call state. New participants in a call session do not have the session membership information, and therefore do not know where to obtain the complete session state. We reject the approach of using a centralized session state manager. This would introduce a single point of failure and a bottleneck in the control path since all state updates must flow through it. Finally the single entity managing call state must also implement rigorous error detection and recovery, incurring protocol complexity and implementation overhead. Rather, we maintain the call state as *soft state* [6]. Soft state is defined operationally as state that is periodically refreshed by update messages. Protocol actions are triggered by new messages and timer expirations. Because the session state is sent to the call session periodically (see Section IV-B for details), newcomers to a session will automatically obtain the current session state through these update messages (with some latency).

The use of the shared group communication channel, combined with a soft state protocol for call state management, maps perfectly onto the *Light Weight Session* (LWS) architecture [33]. LWS is described as a lightweight (soft state), loosely-coupled, and decentralized (anonymous CAs) communication model. We use IP multicast as the shared group communication channel and call sessions are identified by multicast addresses.

Using one multicast address per call session raises two scalability concerns: multicast address scarcity and scaling routing state to a large number of small multicast groups. The first problem, due to the flat IPv4 multicast address space, will vanish with the deployment of IPv6 [4], MASC [18], or with new IP multicast models like Simple Multicast [14] and Express Multicast [19]. For the second problem, by pushing the routing state to edge routers of the backbone [5], or to end systems [28] (such as iPOPs), and building the control structure across these edge routers or end systems through tunneling, the accumulation of the routing state for all the multicast groups in the backbone is prevented, and this yields scalability to a large number of groups.

Our signaling protocol consists of two phases: call session establishment and call session maintenance. The first phase is the formation process of the call session, namely instantiating the CAs, which in turn join the multicast session. The latter phase involves exchanging call states among the CAs in the session, and creates or modifies the data path based on the call states. In [39], we described our signaling approach and call session maintenance in greater detail.

We first describe the call session establishment protocol in Section IV-A, then the call session maintenance and control protocol in Section IV-B.

A. Call Session Establishment

The communication in this phase is pairwise among CAs. Group communication takes place after the session is initially established (Section IV-B).

Figure 2 shows the message flows and state transitions in call state machines on IAPs during a call session establishment. We used the following techniques to provide a fault tolerant call session establishment protocol:

- For all the calls handled by an IAP, the IAP sends periodic, idempotent, and stateful ICEBERG call requests, as their keep-alive heartbeats, to their serving iPOPs. The call requests contain the current state in the call state machines maintained on the IAPs, and are identified by the call party

[9] A call party can use multiple devices at the same time in a coordinated way. If these devices are used in an uncoordinated way, then the call party is merely making multiple calls at the same time.

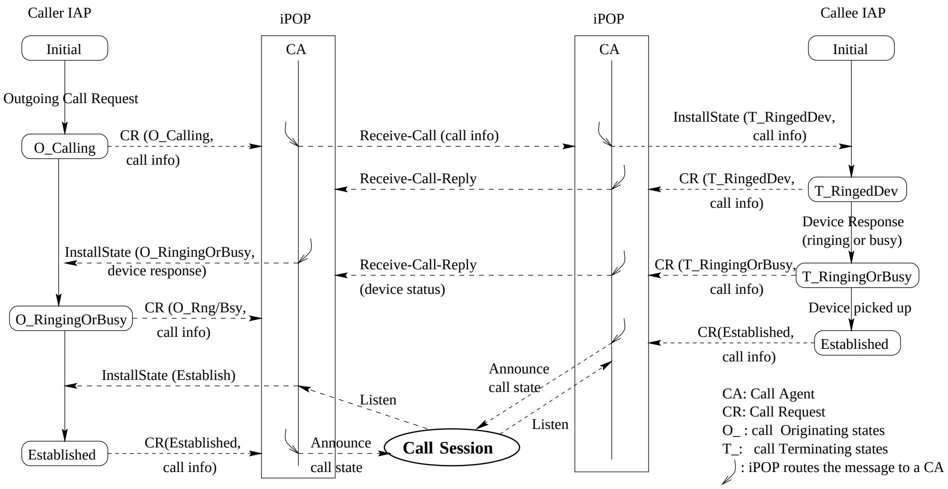
Caller IAP
iPOP
iPOP
Callee IAP
CA
CA
Initial
Initial
Outgoing Call Request
O_Calling
CR (O_Calling, call info)
Receive-Call (call info)
InstallState (T_RingedDev, call info)
Receive-Call-Reply
CR (T_RingedDev, call info)
T_RingedDev
Device Response (ringing or busy)
InstallState (O_RingingOrBusy, device response)
Receive-Call-Reply (device status)
CR (T_RingingOrBusy, call info)
T_RingingOrBusy
O_RingingOrBusy
CR (O_Rng/Bsy, call info)
Device picked up
CR(Established, call info)
Established
InstallState (Establish)
Announce call state
Listen
Listen
Established
CR(Established, call info)
Announce call state
Call Session
CA: Call Agent
CR: Call Request
O_ : call Originating states
T_: call Terminating states
: iPOP routes the message to a CA

Fig. 2. Call Session Establishment

iUID and his/her device in use.

- The call requests are handled by the call state machine-aware (but soft state) CAs. An iPOP maintains a table mapping each call request to its associated CA just the same as in AS1. iPOPs are responsible for routing the call requests and other messages to their serving CAs, or spawning new CAs for new call requests. Based on the call state carried in a call request, the CA performs appropriate actions, such as name lookup or Preference Registry lookup, etc.
- CAs advance the call state machines on the IAPs to the next state by sending periodic "InstallState" messages to the IAPs until call requests with the new state arrive at the CAs.
- Inter-iPOP communication also takes the form of idempotent soft state heartbeat messages for the detection of network partitions between the iPOPs.

The nature of soft state and idempotent messages enables our protocol to recover gracefully from transient failures with no extra logic in addition to the normal operation. In addition, prolonged failures can be detected and the appropriate clean-up actions taken in reaction to them.

Now we illustrate the fault detection and recovery of our protocol in detail. CAs are kept alive by the periodic call requests from IAPs. When a CA fails, the next call request will cause the iPOP to spawn a new CA for the call. The call request contains sufficient state for the new CA to continue the service of the call. A timeout on call requests indicates that either the caller has hung up the call or the IAP is network partitioned from the iPOP. As a result, the iPOP terminates the associated CA, and removes the call request and CA pair from its table.

Timeouts on the heartbeats between the two iPOPs indicate an extended duration of network partition between the iPOPs, and the call establishment is aborted.

iPOPs also send heartbeats to IAPs so that the IAPs can detect network partitions between themselves and IAPs (not shown in the figure). A non-transient network partition causes an IAP to use an alternative iPOP to continue call establishment. The heartbeats between the two iPOPs convey the new identity of the iPOP to the other iPOP. The soft state messages exchanged between the two iPOPs eliminate the need for explicit error recovery on both iPOPs.

B. Call Session Maintenance and Control

Following the call session establishment phase is the call maintenance phase. During this phase the call participants may hang up or interact with their devices in the middle of the call (e.g., invite a new call party, switch to a new device, or perform call forwarding, etc.). This phase involves altering and propagating call states to all the CAs in the established call session. The call state of a call session includes call party identities, the involved communication devices and their status (e.g., busy or on hold), data path endpoint information for all the data streams in the session (e.g., IP addresses and port numbers of the data stream end points, location of transformation agents, etc.), and the time the call state is sent. Maintaining dynamic call sessions and carrying out the control functions in a fault tolerant fashion completes the design of a robust signaling system. In this section, we describe the control protocol in detail.

Each IAP periodically sends the call request with the *Established* state to the iPOP in this phase, as shown in Figure 2. The call request contains the call state of its endpoint. The call state is one of the rows in Table I. The CA, in turn, periodically announces the call state in the call request to the call session. At the same time, the CA also listens to the multicast channel and receives call states of the other endpoints. Hence, each CA maintains the complete call state of a call session: the entire Table I, each entry of which is uniquely identified by "iUID" and "Device" (the primary key of the table). The reliability of call state propagation is ensured simply by periodic retransmission over the lifetime of the call session.

iUID	Device	Status	Path Endpoint Info	Time sent
alice@domain1.com	laptop	busy	123.3.4.5/9876	13:34:45:79
bob@domain2.com	cell phone	busy	gw: 10.3.2.2/66 time slot: 5	13:34:42:87
...	...	...	...	...

TABLE I
CALL STATE IN A CALL SESSION

The combination of periodicity and the use of multicast is often called the *announce/listen* model in the literature, and is appropriate where *eventual consistency* rather than transactional semantics are sufficient. A desirable property of the announce/listen model is that component failures are tolerated as normal operations, rather than addressed through a separate recovery procedure [1]. Recovery is enabled by simply listening to channel announcements. The announce/listen model initially appeared in IGMP [7], and was further developed and clarified in systems such as the MBone Session Announcement Protocol [32].

When a CA receives a new call state (i.e., if the state's key cannot be found in the call state table), it knows that a new endpoint just joined the call session, and this CA needs to establish a data path to the new device through the Automatic Path Creation Service (more in Section V-A). Call state changes (for example, changing the data path endpoint information) are made by the IAPs through periodic call requests with the modified call state, and then the modified state is propagated to the call session through the CA's periodic announcements. When a CA receives a modified call state, the associated data path will be modified accordingly.

Each CA of a session is oblivious to the transient failures and recovery of the other CAs, since the recovered CA only needs to join the signaling multicast group and re-build the call session state (including the additional session state changes made during the CA fault recovery). Prolonged failures of a CA will cause the device it serves to be cut off from the session. Extended network partitions among iPOPs involved in a call during this phase are detected by the call state timeouts, and for a multi-party call, the call may be partitioned into multiple calls. Nonetheless, the partitioned calls will automatically re-integrate when the network partitions are healed.

Despite the simplicity of the design, this scheme handles simultaneous call state changes at multiple CAs gracefully.

We are investigating the appropriate choice for heartbeat intervals. This variable affects the time to recognize network partition and IAP failures.

C. Discussion

The signaling system described above meets the following design goals:

- **Fault Tolerance:** Through the use of lightweight call session and soft state protocols, the signaling system tolerates component failures, adapts to dynamic call session membership and detects network partitions.
- **Scalability:** The use of group communication rather than pairwise communication for signaling scales to a large number of call participants in a call.
- **Network and Device Independence:** Our signaling protocol is designed to be independent of the access networks and devices.

V. Service Creation

Easy service creation is an essential goal of the ICEBERG architecture. The concept of a data path is an enabler and simplifier for service creation, which we discuss in Section V-A. Our signaling protocol primitives and other ICEBERG components, such as the Preference Registry, Name Mapping Service, Personal Activity Coordinator also empower novel services and simplify their creation process, which we illustrate in the following sections.

A. Data Path, a Simplifier

Data path construction establishes the flow of data between the communication endpoints of different call parties. A path consists of a sequence of operators and connectors (see Section III). The power of the path concept comes from the flexible composability of operators and connectors. Given two endpoint data formats and locations, the APC service automatically construct a path consisting of a sequence of operators with appropriate connectors and maintains the path in a fault tolerant fashion by restarting failed operators and connectors. We do not yet have a full-fledged Automatic Path Creation Service. The current APC service only creates and places predetermined operators at the appropriate locations. An operator description is associated with each operator for its creation (which code to execute), placement (which host in the cluster), and destruction (what to clean up after the operator terminates). Modifying a data path involves changing the operator descriptions, and restarting or modifying the currently executing operators.

Creating paths in the wide area and maintaining them in a fault tolerant fashion are the main challenges for data path creation and maintenance. A centralized APC service responsible for instantiating, executing, and maintaining the paths is not appropriate for the wide area, because when it is out of service (from host crashes or network partitions), it orphans all the paths it maintains (with operators and connectors running on different hosts). In addition, the reality of the multiple independent service providers precludes the practicality of using a centralized APC service.

Distributed path creation and maintenance pose difficult questions of path ownership and access control (i.e., who is allowed to create or destroy paths). Also, our signaling protocol is based on an asynchronous (soft state), anonymous[10] (using multicast address as a level of indirection and the rendezvous) group communication model, which makes the protocol incompatible with synchronous path creation and maintenance between pairs of endpoints. Multiple owners of a single path would need to cooperate synchronously to create and destroy the path, so a path should be owned by only one entity. This is somewhat counterintuitive since a path describing the communication between two devices does indeed involve two endpoints, and therefore, the path is perceived to be owned by these two endpoints.

Instead, we present a different view for this scenario with the following additional information: an intermediate data format (that may coincide with the data format supported by either endpoint). From each endpoint's view, all it needs to do is to construct a data path to send and receive data using the intermediate data format. This essentially decomposes the communication between two devices into two paths with the output data format of one path the same as the input data format of the other, namely the intermediate data format. This way, each endpoint owns its half of the path, and it creates and destroys the data path autonomously with no need to synchronize with the other end-

[10] Anonymity here refers to the fact that any CA is oblivious to dynamic session membership, and does not mean that anyone can participate the call session. In fact, call participation is invitation based, and *not* subscription based.

point. Each half-path can be composed of operators all running on the same iPOP that is serving the endpoint, so it is reasonable to run a single APC service on each iPOP (where network partitions are rare), responsible for creating half-paths for endpoints serviced by that iPOPs. Like any Ninja service, the APC service will be highly available and fault tolerant.

The intermediate data format is determined in the following manner. Each iPOP knows what operators and connectors it has and which are running. Therefore, it can export a list of data formats that it supports. During the call setup, the caller's iPOP obtains this list from the callee's iPOP. Then it selects a matching data format as the intermediate data format. We currently take the first match, but we are investigating more sophisticated algorithms that select the most cost-effective data format for all call participants.

Now we address the failure modes of data paths. Data paths are separated from control paths, except at an IAP, since the network-specific gateway interacts with both the control and data components of the access network. Thus, the failure of the IAP will jeopardize the data path as well. However, failures (although unlikely) of Call Agents, Name Mapping Servers, and Preference Registries will not affect the data path. Failed operators and connectors in a data path are restarted by the APC service running on the Ninja Base.

B. Multi-device Call Primitives for Easy Service Creation

The operations that carry out the basic call service are the primitives for additional services. By having multi-device communications as the basic call service, we have enriched the set of the building blocks. The multi-device call operations are adding a new communication endpoint of any type to the call and removing an existing communication endpoint from the call. Changing a communication endpoint adds the new endpoint, and then removes the existing one. Services that require endpoint changes, such as call forwarding and call transfer, can be implemented easily on top of these *user-initiated* operations. For example, call forwarding can be implemented by having the forwarder invite the forwardee endpoint and then leave the call.

Service handoff is another example. It occurs when users change their communication devices during a call. Service handoff enables service mobility, the ability to retain access to services as users switch between networks. In telecommunications, the term "service mobility" usually means "service portability", the ability for diverse networks to share user service profiles and to carry out the same set of services in each network [15]. However, when one crosses the network boundary in the middle of a service session, that service is terminated. The user must then restart the service in the new network. In contrast, service mobility provides *seamless* integration of heterogeneous devices from diverse networks, as if they are accessing the same network, achieving the goal of *Potentially Any Network Service* (PANS).

Service handoff can also be viewed as a generalized call transfer service. Instead of call transferring between homogeneous telephones, service handoff allows one to perform call transfer across diverse devices from heterogeneous networks.

Now imagine the following service handoff scenario: Alice is on her cell phone. She walks into her office and decides to handoff the call to her laptop IP phone. She does this by first pressing some keys on her cellular phone to indicate the intent and the target of a service handoff (e.g., "*SHIP" – Service Handoff to IP phone). The serving IAP that contains a cellular gateway intercepts the DTMF (Dual-Tone Multi-Frequency) or touch-tone message and relays it to the serving Call Agent for Alice's cell phone. The Call Agent looks up the end point information (the IP phone's IP address and port number in this case) in the Preference Registry with the input of Alice's iUID and endpoint type ("IP"). Then the Call Agent invites the IP phone to join in the conversation, as illustrated in Section IV-A (i.e., the operation of adding a communication endpoint to a call[11]). Finally, Alice turns off her cell phone, which informs the Call Agent that was serving the cell phone that it should leave the call (i.e., the operation of removing an endpoint from the call) and completes the service handoff.

C. ICEBERG Components as Service Enablers: Our Experience

C.1 Universal In-box

The Universal In-box is a service that enables the user to receive or retrieve voice mail, e-mail, phone-calls, fax, instant-messages, etc. in a unified fashion. It features an any-to-any communication capability between the different endpoints (e-mail to fax, voice-mail to e-mail, pager-message to phone-call, etc.). This application was one of the first we built on top of ICEBERG and it drove much of the initial design. It exemplifies the *device independence* and *personalization* aspects of ICEBERG. Device *name independence* is provided by the Name Mapping Service. Device *type independence* is enabled by the IAP (which does the protocol conversion) and the APC service (which takes care of any data format conversion in a generic fashion).

The Preference Registry provides a clean model for the personalization features of the Universal In-box. We have built the *Preference Manager*, a graphical tool that allows users to conveniently specify their communication handling preferences for the Universal In-box.

The Universal In-box started with support for just two endpoints: cell-phones and *vat* (desktop audio tool). We have since added support for voice mail, e-mail and recently, interfaced to an instant messaging system. The no-frills interface to the instant messaging system, which is less than 300 lines of Java code (with the appropriate calls to the other ICEBERG components) was half a day's work. Although we did not have a fully functional APC service during these additions, we have found adding a new data format and access network to be straightforward. This simplicity is due to the clean separation between the ICEBERG components which provides different levels of independence. Thus, this application has given us good experience in building services that are extensible to new endpoints.

C.2 Jukebox Service on ICEBERG endpoints

The Jukebox service was built as part of the Ninja project [11], independent of ICEBERG. It plays MPEG3-encoded songs stored on the disks in a cluster of workstations.

As an exercise in testing the ease of introducing new services in ICEBERG, we have built a virtual IAP (with no actual access network gateways, but similar to a proxy) for the Jukebox

[11] A data path is not created between two devices of one call party.

service that makes the service behave as an ICEBERG communication endpoint. This IAP handles calls from any ICEBERG endpoint to the Jukebox service and maintains the call state machine on behalf of the Jukebox. The special functionality in this IAP is its ability to interpret requests issued to the Jukebox from the caller, such as the name or ID of the song (this could be entered as DTMF signals from the caller's device such as a cell phone). Then these requests are translated to regular Jukebox service requests for the Jukebox service.

The ease of introducing new services is demonstrated by the fact that building the single IAP was sufficient to make the service available to *any* ICEBERG endpoint. Implementing a no frills IAP required only 500 lines of Java code, most of which implemented the proxy functionality.

The Jukebox service is now available through this IAP to the endpoints in our testbed - GSM cell phones and IP desktop audio applications, *vat*. We expect that as we add other types of endpoints to our testbed (like the PSTN phone), the service will be immediately accessible from those endpoints as well.

D. Room Control Service

We have implemented a complex composable service for multi-modal control of smart spaces using several of the ICEBERG access networks. The end-devices we connected to the system include a desktop computer GUI, microphone and speakers, a cell phone, and a 3Com Palm Pilot Personal Digital Assistant. Users can use graphical, text, or speaker-independent speech-based user interfaces to interact with one another and to control the A/V resources in several rooms in Soda Hall.

The Smart Spaces Control (SSC) server uses the Automatic Path Creation service to create a path between the sender and recipient that performs any necessary transcodings (e.g., transcoding GSM-encoded audio to PCM-encoded audio or performing speech-to-text recognition). Using a control channel that is parallel to the data channel, we can dynamically customize the operators (e.g., the speech recognizer uses a constrained n-gram, based upon room control commands, to improve recognition accuracy and speed).

The implementation of the SSC applications leveraged the ICEBERG and Ninja infrastructure and was completed with only 1000 lines of code in only a few person-months of work. Extending the applications has been equally easy. Adding support for a graphical Personal Digital Assistant, using a GUI instead of speech, required only two hours of work.

We are currently extending the speech recognition operators to include more sophisticated operators, such as pause removal, pitch emphasis detection, time code extraction, and confidence extraction.

VI. Clearing House: Resource Allocation and Billing

A. Overview

In this section, we describe our initial design of the Clearing House (CH), which coordinates the interactions between the ICEBERG Network plane and the ISP Plane (Figure 1). The structure of CH is designed for efficient provision of resource reservations and secure billing of services. In addition, the CH must scale over both distance and number of ISPs to support the operation of ICEBERG in the wide area.

To make the CH scalable to a large user base over a wide area network, we are implementing the CH as a hierarchical distributed system. We assume the network to be composed of various basic domains (based on either administrative or geographic boundaries), with minimal domain overlap. In our design, physically adjacent domains are aggregated to form bigger *logical domains*. This induces a hierarchy of *logical domains* in the network and a distributed CH is associated with each of them. At each level, multiple CH's share the task of searching through the ISP space to construct inter-domain subpaths, as well as storing this information for future usage. When a call between two endpoints across the wide area is requested, a recursive call will be made from the parent level of the CH hierarchy for a path between a local ISP and an adjacent domain. The children CH nodes will perform the local search automatically to construct paths which span to the boundary of the adjacent domain.

The implementation of the CH is still on-going, and further studies are needed to map the CH hierarchical tree and its logical domains to the existing Internet that spans multiple administrative domains without clear geographical boundaries.

B. Clearing House Infrastructure

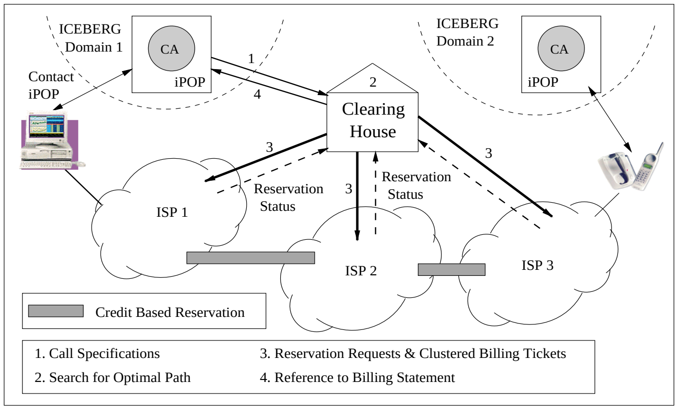

Fig. 3. A High-level view of the CH architecture during call setup.

First, we demonstrate how the CH handles the various tasks by describing a typical sequence of events that take place after a call setup request arrives at the iPOP, as labeled steps 1-4 in Figure 3:

1. **iPOP passes call specifications to the CH.**
   When the iPOP receives a new call request, it looks up the Name Mapping Service and PAT to locate the caller and the callee, and obtains their preferred contact point from their respective PR (Section IV-A). The iPOP then contacts the CH, and passes along the following call specifications: *[caller location, callee location, end device of contact, traffic statistics, desired quality of service, misc.]*
2. **The CH performs an optimal path search.**
   The CH uses its internal cache information about inter-ISP reservation status and the call specifications to search for the optimal end-to-end packet flow path through ISPs. This path is optimized based on the desired quality of service (e.g. network latency), and reservation availability.
3. **The CH sends reservation requests and billing tickets to ISPs.**
   The CH aggregates reservation requests and billing tickets

for calls that travel through the same ISP. After a preset time period, the clustered reservation requests and billing tickets are sent to each of the ISPs involved in the path.

4. **The CH authorizes the ISPs in the path.** If sufficient resources are reserved along the optimal path, the call is admitted.

Admission Control: If the CH fails to locate any links with sufficient resources reserved to complete a chosen path, the CH will either block the new call, or renegotiate with the iPOP for the amount of required resources to carry the call. The decision involves some trade-offs in the user perceived quality of the call, the cost for increasingly scarce resources, and the additional latency to complete the call setup.

In the following discussions, we elaborate on how the CH provides resource reservation and secure billing services to the ICEBERG architecture.

B.1 Scalable Resource Reservation

We propose a credit-based resource reservation scheme, of which one very important feature is that the resource reservation is not performed on a per-call basis, but rather is aggregated over various connections that use a particular link. As shown in Figure 3, resource reservations are set up from ISP to ISP, instead of end-to-end. The neighboring ISPs send regular updates to the CH which keeps track of the ”reservation status”, i.e., how much of the available resources are currently reserved on a particular link. The truthfulness of such updates can be verified by companies such as Inverse Network Technology [29]. An ISP pays another ISP a certain amount of credit to reserve resources for a preset time window, and one can enhance the reservation by increasing the credit of the existing reservation, instead of requesting a separate reservation.

Since online resource reservation is very costly and time consuming, the goal of our design is to minimize the amount of per-link reservation that has to be done during call setup for a particular call. In Step 2 above, the optimal path is chosen such that the number of new per-link resource reservations is minimized.

B.2 Billing

Once the CH finds the optimal path, it returns an authorization ticket and a billing ticket for each ISP involved to the iPOP. For each aggregate reservation request, the CH sends the authorization tickets in groups to each ISP. Along with this, the CH also periodically sends the clustered billing tickets to each ISP.

The iPOP hands the pairs of tickets to each ISP on the path, setting up the call in the process. Each ISP authenticates the authorization ticket before allowing the call setup. After each call ends, its billing ticket is kept in persistent storage. At the end of the regular billing cycle, each ISP aggregates its billing tickets and sends them to the CH for lump settlement payments.

Security: We assume that the ISPs trust the local CH to store their service information. The CH's security structure must prevent ISPs from forging their own billing tickets or unauthorized call setup messages. We provide a secure system that uses digital signatures and asymmetric and symmetric encryption to provide security while maximizing performance (see further discussions in Section VII). To reduce the CPU overhead imposed on the CH by encryption operations, the CH aggregates multiple billing tickets for a given ISP within a fixed time window, and provides a digital signature for the concatenated billing tickets.

B.3 Discussions

The resource reservation and billing mechanisms described above meet the following design goals:

- **Scalability:** Resource reservation requests are made for aggregate connections instead of for individual users. Neither the CH nor the ISPs need to maintain per connection state. Similarly, the CH clusters the corresponding billing tickets for a preset time window before sending them to ISPs. This allows the CH to scale nicely.
- **Heterogeneity:** The CH design provides resource reservation and billing mechanisms that operate across heterogeneous access networks and services. Since reservation is done using time-based credits, service handoffs can be performed without having to tear down resource reservations on every link. This functionality supports PANS and personal mobility in ICEBERG.
- **Minimize Signaling Overhead and Setup Time:** The flexibility in allowing existing resource reservations to be enhanced greatly reduces the need to set up new reservations for every single link whenever a new call is made. Since reservations may already exist on some links in the sender-to-receiver path, both resource reservation and call setup time can be reduced. The clustering of billing tickets also reduces the amount of CPU overhead for encryption operations, and lessens the number of control messages that are exchanged.
- **Inter-operability in the Wide Area:** The CH's design can easily be extended to work with future Internet protocols that provide different service levels (e.g., Differentiated Service or Integrated Service with RSVP).

VII. Security and Authentication Issues in Call Setup

ICEBERG is built upon the untrusted Internet, which poses privacy and authentication issues in the call setup process. We briefly discuss these issues in this section.

Authenticating Name Mapping Service lookup: This lookup operation is an important step in call setup. Name servers in our architecture are akin to DNS name servers: they are assumed to store public information. This abstraction simplifies the authentication requirement for a Name Mapping Server (NMS) lookup — only the reply from the NMS needs to be authenticated, and the reply need not be encrypted. We use the Name Mapping Service to bootstrap authentication; it is used to get the public key of any user or network infrastructure component that is required for further cryptographic operations. We assume the existence of a PKI (Public Key Infrastructure) for authenticating the Name Mapping Service reply.

Preventing caller spoofing: Call setup (from caller A to callee B) involves the lookup of B's Preference Registry (PR_B) by A's Call Agent (CA_1). At this stage, PR_B has to verify that CA_1 is a valid Call Agent of the caller to prevent bogus callers. In our model, PR_A, user A's Preference Registry, is considered to be the authority of the information about the list of valid Call Agents for A. This model is flexible: PR_A can update the list when the user subscribes for the services of CA_1 and informs

PR_A about it (through an out of band channel). PR_A can then issue a certificate to that effect to CA_1, which can then be presented to PR_B for the required authentication.

Enforcing callee control and callee privacy: There is another subtle requirement in the call setup process when the originating and terminating call agents communicate ($CA_1 \rightarrow CA_2$): CA_2 has to be able to verify that CA_1 has indeed contacted PR_B in the recent past. This is to enforce callee control and disallow callers from by-passing the Preference Registry before calling. Our approach to deal with this is to have PR_B issue an encrypted, signed ticket to CA_1. This ticket, visible only to PR_B and CA_1, is for one-time use only, and has an expiration timestamp. The ticket is used by CA_2 for the required verification. Call privacy is also supported by this mechanism: PR_B can hide the actual endpoint identity of the callee by encrypting this information in the ticket.

Secure billing: To prevent tampering of billing tickets, we need to provide authenticity guarantees. The natural choice is to use public-private key encryption, where the CH digitally signs a billing ticket with its private key for authenticity. Using the digital signature, a provider can authenticate the origin of the billing ticket, and provide undeniable proof of the charged amount. However, public key cryptography imposes a much higher CPU overhead than symmetric key encryption. Our design choices directly impact call setup latency, and require further studies of the trade-offs involved.

Encryption for secrecy in any of the above steps is done in one of two ways: (a) by using the public key of the recipient, or (b) by establishing a shared session key before information exchange. Both these methods could involve extra round-trips (for learning the public key, or for establishing the session key). The latter method (established session key) would be used for secrecy of the messages exchanged during call session maintenance. Note that the use of IP multicast makes call session management vulnerable to denial-of-service attacks. Nonetheless, new multicast proposals, like Simple Multicast [14], offer access control and prevent denial-of-service attacks.

The complete call setup process could involve a significant amount of latency for all of the authentication steps. Fortunately, almost all of the authentication steps can be optimized by caching previous authentication information, public key lookup information, or the pre-established shared session keys. These optimizations can save network round-trips, as well as cryptographic operations at the endpoints.

We are in the process of implementing these security mechanisms in our existing ICEBERG testbed.

VIII. Implementation

To gain experience and to iterate on our design, we have been implementing ICEBERG components in a testbed. This currently consists of a GSM cellular base-station, laptops with WaveLAN wireless interfaces, a H.323 gateway, and a two-way paging base-station. We have built IAPs for interfacing with cell-phones and IP-telephony, and are in the process of building IAPs for the H.323 and paging gateways.

In terms of the other software components, we have implemented our signaling protocol in Call Agents, the Preference Registry, the Name Mapping Service, and the APC service. We are using the H.323 gateway in our testbed to experiment with billing mechanisms. We are in the process of working out the next level of details of our Clearing House design and security design.

The Call Agent, Preference Registry, and the APC service are implemented as services in Java with an RMI interface. We're currently using an LDAP server (from www.openldap.org) for service location and the Name Mapping Service.

Our implementation is as yet untuned and shows that a high-end PC (with 500Mhz Intel Pentium II processor and 256MB RAM) can handle 10 call initiations per second. The current telephone usage model, according to [30], is a mean call arrival rate and a mean call duration during busy hours of 2.8 calls per hour and 2.6 minutes per call respectively. Using this model, we would require approximately 500 PCs to handle the call traffic for a region on the scale of the San Francisco Bay Area with a population of six million.

Our examination of the call processing latencies shows that a large part of the cost is due to the Java services. RMI lookup and calls take on the order of 10ms and require multiple network round-trips. Performance is also limited by the fact that we're using a Java implementation with user-level threads that spin idly while waiting for network packets. With a large community of researchers working on Java CPU, as well as, Java I/O performance, we expect that these costs will be reduced.

IX. Conclusions and Future Work

In this article, we presented the ICEBERG network architecture: an Internet-core network for integrated communications. Our network can be viewed as islands of cluster computing platforms that offer scalable, available, and robust services on them. Our signaling protocol takes the approach of lightweight sessions and soft state, enabling scalable and robust call services in the wide area. With multi-device communication as the basic call service combined with ICEBERG components that manage user preferences, behaviors, and compose units of computation, we provide an easy service creation environment for customizable services. To address wide area quality of service issues, we use a Clearing House-based approach that leverages aggregation to provide scalable resource reservation and billing mechanisms. The ICEBERG architecture also provides secure, authenticated call setup and billing for services.

We have a significant amount of future work ahead of us, in particular we have not yet addressed the problem of Operations, Administrations and Maintenance (OA&M), a critical technology that is mature in modern telecommunications networks. In addition, we are continuing our experiments with more sophisticated and novel services and we are in the process of formulating a well-defined service creation model. We also plan to address the incremental wide area deployment of ICEBERG architecture and Clearing House infrastructure over the existing Internet.

References

[1] Elan Amir, Steven McCanne, and Randy H. Katz. An active service framework and its application to real-time multimedia transcoding. In *Proceedings of ACM SIGCOMM 98*. ACM, August 1999. Vancouver, British Columbia.

[2] T. E. Anderson and et al. The Case for NOW (Network of Work Stations). In *IEEE Micro*, feb 1995.

[3] N. Anerousis and et. al. Tops: An architecture for telephony over packet networks. *IEEE Journal of Selected Areas in Communications*, jan 1999.

[4] W. Biemolt, M. Kaat, and H. Steenman. *A Guide to the Introduction of IPv6 in the IPv4 World.* Internet Engineering Task Force, April 1999.

[5] Ljubica Blazevic and Jean-Yves Le Boudec. Distributed Core Multicast (DCM): a multicast routing protocol for many groups with few receivers. In *NGC*, Pisa, Italy, nov 1999.

[6] D. D. Clark. The design philosophy of the DARPA Internet protocols,.

[7] Steve Deering. *Host Extensions for IP Multicasting*, Aug 1989. IETF RFC-1112.

[8] B. Stiller et al. Charging and accounting technology for the internet. In *Multimedia Applications, Services, and Techniques*, May 1999.

[9] Guido Appenzeller et al. The Mobile People Architecture. In *Technical Report of Stanford Computer Science Lab CSL-TR-99-777*, January 1999.

[10] N. Duffield et al. A flexible model for resource management in virtual private networks. In *ACM SIGCOMM*, September 1999.

[11] Steven D. Gribble et al. The MultiSpace: an evolutionary platform for infrastructural services. In *Usenix Annual Technical Conference*, June 1999. Monterey, CA.

[12] C. Gbaguidi et.al. An Architecture for the Integration of Internet and Telecommunication Services. In *IEEE Infocom '99*, New York, March 1999. IEEE.

[13] Jean-Pierre Hubaux et.al. The impact of the internet on telecommunication architectures. In *The International Journal of Computer and Telecommunication Networking*, 1999.

[14] R. Perlman et.al. *Simple Multicast: A Design for Simple, Low-Overhead Multicast*. Internet Engineering Task Force, December 1998.

[15] Nadege Faggion and Cegetel Thierry Hua. Personal communications services through the evolution of fixed and mobile communications and the intelligent network concept. *IEEE Network*, Jul/Aug 1998.

[16] A. Fox and et al. Scalable network services. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP-16)*, St. Malo, France, oct 1997.

[17] M. Günter and T. Braun. Evaluation of bandwidth broker signaling. In *IEEE Seventh International Conference on Network Protocols (ICNP)*, October 1999.

[18] M. Handley. Session Directory and Scalable Internet Multicast Address Allocation. In *Proceedings of ACM SIGCOMM*, sep 1998.

[19] Hugh W. Holbrook and David R. Cheriton. IP Multicast Channels: EXPRESS Support for Large-Scale Single-Source Applications. In *Proceedings of ACM SIGCOMM*, sep 1999.

[20] http://info.ox.ac.uk/fax/. *Email to Fax at Oxford University*.

[21] http://ninja.cs.berkeley.edu/. *The Ninja Project*.

[22] http://planetarymotion.com/. *Planetary Motion's CoolMail Service*.

[23] http://www.databeam.com/h323/h323primer.html. *A Primer on the H.323 Series Standard*.

[24] http://www.fax away.com/. *Faxaway: The Premier Email to Fax Service*.

[25] http://www.internet2.edu/qos/qbone/. *Internet2 QoS Working Group: Qbone testbed*.

[26] http://www.thinklink.com/. *ThinkLink*.

[27] http://www.wildfire.com/. *Wildfire*.

[28] Yang hua Chu, Sanjay Rao, and Hui Zhang. *EndSystem Multicast*. http://www.cs.cmu.edu/ kunwadee/research/other/endsystem-index.html.

[29] Inverse network technology. http://www.inverse.net/.

[30] C. N. Lo, R. S. Wolff, and R. C. Bernhardt. An estimate of network database transaction volume to support universal personal communications services. In *First International Conference on Universal Personal Communications (ICUPC '92)*, 92.

[31] Colin Low. The internet telephony red herring. In *Proceedings of Global Internet*, November 1996. London, England.

[32] Maryann P. Maher and Colin Perkins. *Session Announcement Protocol: Version 2*, nov 1998. IETF Internet Draft: draft-ietf-mmusic-sap-v2-00.txt.

[33] Steven McCanne. Scalable multimedia communication with internet multicast, light-weight sessions, and the mbone. Technical Report CSD-98-1002, Computer Science Division, U.C. Berkeley, july 1998.

[34] British Columbia Institute of Technology. *CA*net II Differentiated Services: Bandwidth Broker High Level Design*, November 1998. http://www.merit.edu/working.groups/i2-qbone-bb/.

[35] Mema Roussopoulos and et al. Personal-level routing in the mobile people architecture. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, oct 1999.

[36] H. Schulzrinne and J. Rosenberg. A comparison of sip and h.323 for internet telephony. In *Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, July 1998.

[37] Henning Schulzrinne. A comprehensive multimedia control architecture for the Internet. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video*, May 1997.

[38] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: an authentication service for open network systems. In *USENIX Association Winter Conference*, Dallas, TX, 1988.

[39] Helen J. Wang, Anthony D. Joseph, and Randy H. Katz. A signaling system using lightweight call sessions. In *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[40] Mohammed Zaid. Personal mobility in PCS. *IEEE Personal Communications*, Fourth Quarter 1994.