Achieving Service Portability in ICEBERG

Zhuoqing Morley Mao, Randy Katz
{zmao, randy}@eecs.berkeley.edu
CS Division, EECS Department, University of California at Berkeley, CA 94720, USA

Abstract—**There is a growing trend toward service access through heterogeneous devices from diverse networks. It thus becomes crucial to provide support for mobility and portability at the application and service level to enable seamless service access from any end devices and access networks. Such support allows transparent roaming and ubiquitous service access. In the ICEBERG [1] project, our goal is to develop such a service infrastructure to integrate a variety telephony and data services spanning diverse access networks reaching heterogeneous end users. In this paper, we discuss our techniques for achieving goals of personal and service mobility, transparent network- and device-independent service access, as well as highly scalable and fault-tolerant access to composed service entities across the wide area. We evaluate our implementation through applications such as Universal In-box, Interactive Voice Room Control, MP3-Jukebox access using a cell-phone, and real-time video delivery to wireless clients.**

Keywords— **Service composition, service mobility, personal mobility, resource optimization, service scalability, fault-tolerance, service personalization**

I. Service customization using paths

The goal of the ICEBERG [1] project is to develop an Internet-based integration of telephony and data services spanning diverse access networks. We define *Service portability* to be the ability to access services using any devices, anywhere, continuously with mobility support and dynamic adaptation to resource variations. We first describe our motivating scenario that drives the design of our system. Alice wakes up in the morning and is reminded by her Palm that today is her mother's birthday. She fires up a video conference session to wish her Mom happy birthday. She sits down in front her desktop machine with an attached Web camera and picks up her home PSTN phone which rings as soon as her Mom answers. The video stream is automatically directed to the large desktop PC monitor. During the video conferencing call, she is reminded again by her Palm that she is running late for an appointment, thus she pushes the button on her home phone to transfer the call session to her cell-phone. She also transfers the video streams to her Palm with CDPD network connectivity. Since CDPD has a much lower bandwidth, the image size and quality is automatically reduced. The video input is received by the camera attached to the Palm. After completely transferring the video conference session, Alice walks into her garage and gets into her car equipped with a car cell-phone and a CDPD network interface. She puts on the headphone set and docks her cell-phone to transfer the call session to the car cell-phone. The video stream is then automatically redirected to the dashboard display screen.

The key enabling component of such service customization in the ICEBERG architecture is the middleware infrastructure service – **Automatic Path Creation (APC) service** [2]. APC automatically adapts the service output data to the end user's device and current network conditions by establishing the proper data flow, which we call a **path**. A data path consists of a sequence of operators joined by connectors. **Operators** are Internet service instances and computation units responsible for data adaptation through format transformation. Examples are forward error correction (FEC) operators to adapt data to wireless clients and PCM-to-GSM converters for GSM cell-phone users. **Connectors** are an abstraction of the Application Data Unit (ADU) [3] transport mechanism between two operators. They encapsulate properties of the transport such as reliability, order of delivery, buffering mechanism, and drop policy. The separation between computation and the data transport mechanism creates clear fault boundaries, facilitates optimization of resources, and enables component reuse. We achieve automatic composition of operators by mandating services to be strongly typed. Thus, type checking can overcome the semantic information complexity and enable graph search algorithms to easily discover semantically correct and high performance service compositions.

A. Any Device to Any Service

APC allows services to be accessed transparently from any device and any network using its data format transformation capability. Given a mismatch between data formats, APC locates the necessary format transformation operators and intelligently inserts them in the data stream. For instance, a real-time streaming MP3 Jukebox service can easily reach GSM cell-phone users using a data path consisting of MP3-to-PCM and PCM-to-GSM transcoding operators created by the APC service. Another example of seamless service access is retrieving map information from Yahoo! map service using a GSM phone. The data path established converts the HTML format to the GSM format by going through content extraction, speech synthesizer and GSM encoding operators (shown in Figure 1). This technique generalizes to any legacy Web services.

APC service is completely transparent to the user and *application service providers* (e.g., Yahoo) and only interacts with the *Network Service Provider*. For example, when a GSM cell-phone user requests a Yahoo Map service, his GSM Service Provider will detect the format mismatch and request APC to build a data path.

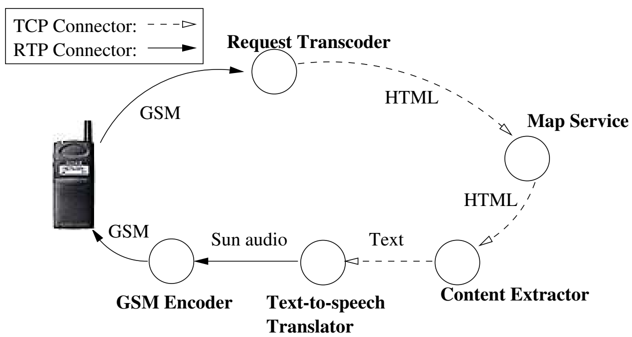

Fig. 1. **Example path use scenario:** This figure illustrates the two paths created for requesting map directions using a cell-phone from a Web map service. Each circle denotes an operator which can be either a long-lived service instance (e.g., Map Service) or a dynamically created one (e.g., GSM Encoder). Each arrow in the figure indicates a connector and the direction of data flow.

The format mismatch between client and service frequently occurs due to client's lack of software or hardware capability as well as other resources such as memory and power. For example, a user with only immediate access to a Real Player can now access a MPEG video streaming service given the data path constructed by APC. Similarly, thin PDA clients with limited display, computational, memory, or power capabilities can conveniently use any existing legacy services without having specialized services built just for them. A direct consequence of on-demand content adaptation is faster service deployment and ubiquitous service access.

B. Enabling Personal Mobility

In addition to format conversion, APC also provides personal mobility support needed by mobile clients. A roaming client switching from the wired network to wireless access may change its IP address but still continues the ongoing service session without any interruption. A session service redirection proxy is provided by the mobile operator created in the data path to dynamically detect user's mobility pattern change and redirect the service data to the mobile clients. Such a redirection proxy also caches application-specific data during temporary network disconnectivity to provide the illusion of a continuous session.

Furthermore, APC enables users to handoff across different devices during a service session. In our motivating scenario, Alice chooses handoff the video stream from her desktop to her Palm Pilot. APC dynamically creates a new data path to transparently redirect the data stream from the service to the new device.

C. Dynamic Adaptation to Resource Variations

Frequently end users may experience degraded performance of the service due to dynamic changes in network conditions, e.g., sudden drop in bandwidth due to network congestion, or other resources such as computational cycles and memory space. There is a need for applications to adapt to dynamic changes in available resources to optimize user's perceived quality of service. We define three ways to adapt to changes in resources.

- **Application-intelligent adaptation:** The application is powerful enough to do its own adaptation to resource changes. For instance, RealAudio combines multiple streams encoded for different bit rates into a single clip, and RealVideo uses a single codec to encode data for all bandwidths. During runtime, the audio and video streams dynamically adapt to changes in bandwidths [4]. In this case, APC directly takes advantage of the application adaptation mechanisms in the composed path.
- **Application-specific adaptation:** The application provides mechanisms for dynamic adjustment, but does not do so automatically. For instance, for bandwidth adaptation, there are different instances of the same codec intended for different bit rates. A codec may be error-resilient, but needs to be notified of the current error rate through a control channel. In this case, APC is responsible for monitoring the resource changes and providing the feedback to the applications to enable dynamic adaptation. APC continuously monitors resource changes and the quality of output data to the client to detect the need for dynamic path optimization.
- **Application-independent adaptation:** If there is a lack of knowledge of the underlying implementation of the application, APC treats it as a black box and does application-independent adaptation. For instance, to adapt to high packet loss rate, forward error correction (FEC) and compression operators are inserted for better data throughput. FEC can also vary the amount of redundancy based on the packet loss rate.

Combinations of the above approaches can be used if needed. Given the path requestor's optimization criteria, APC strives to create service compositions that best utilize network and computational resources to achieve the optimal desired QoS. Optimized resource utilization and differentiated QoS are enabled by our iterative data path construction process with continuous feedback (see Section II) and clear specification of optimization metrics.

D. Enabling Wide-area Service Scalability and Availability

Services need to scale well given increasing load and degrade gracefully under overload. High availability is an important requirement of today's network services given increasing dependence on immediate information access. Traditionally, services are constructed using a thread-based programming style. In ICEBERG, we take advantage of event-driven programming model and asynchronous, nonblocking I/O library to achieve greatly improved scalability through reduced thread overhead in memory usage, context switching, and unnecessary blocking due to I/O. The Automatic Path Creation Service itself is constructed using this programming paradigm to increase path creation throughput.

Moreover, event-driven style exposes the task queue to the application writers to allow application-specific priority request scheduling. Another advantage of this programming style is that it eliminates coupling and blocking effects within the service composition chain and thereby greatly increases the performance of the composed service entity, making services usable under heavy load. When one service composes with another one, the interface of asynchronous task requests and completion replies instead of blocking RMI calls free up requesting service's threads to work on other computations.

Another strategy we deployed for increased service scalability is the use of cluster computing platform for constructing our services. We leverage several nice properties of such a platform: incremental scalability, fault-tolerance, high availability through redundancy, and high network bandwidth [5].

E. Achieving Fault-tolerance for Composed Services

Whenever possible, APC should protect users from the failure of individual path components or communication links between them. It strives to provide the illusion that the user is accessing a single robust service entity providing the composed functionality.

Despite inherent redundancy and low probability for network partition within a cluster, failures can still occur at both the process and machine level due to hardware problems or software bugs. It is critical for a path to gracefully and quickly recover from failures. The APC service is a wide-area, cluster-based service. Thus, it is highly available and fault-tolerant due to redundancy in the cluster. New APC service instances are created in response to failures or high load. On each cluster, multiple APC service instances exist and provide local area fault-tolerance. Across the wide area, APC service instances on different clusters coordinate to build a wide area path show in Figure 2.

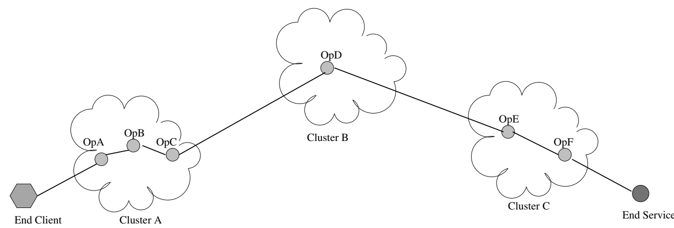

Fig. 2. **An example wide area path:** This figure illustrates an example wide area path. Each circle denotes an operator. Each connecting line between the operators indicates a connector. The path is requested by the end service.

APC provides two redundant sets of control paths for detecting and handling path component failures as illustrated in Figure 3 as part of the run-time environment of the data path.

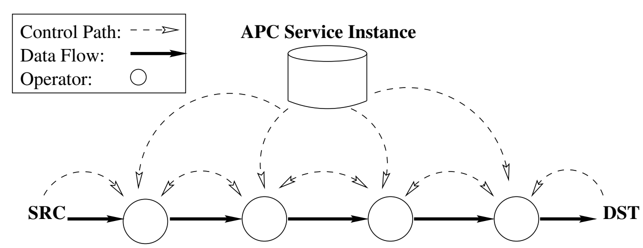

Fig. 3. **Redundant control paths:** To ensure a fast and robust fault-recovery model, multiple control paths are built into the system to guarantee their robustness.

- **APC Monitoring**

APC periodically sends heartbeat beaconing messages to each operator within the path to make sure they are functional. Upon timeout, APC assumes either process or node failure and attempts to restart the operator. First it tries to reuse the existing node provided it is not overloaded. If that fails, APC locates a new node through the cluster. If that is still unsuccessful, reconstruction of logical paths is attempted.

- **Peer Monitoring**

Each operator is network I/O intensive and constantly receives from and sends data to neighboring operators. Upon catching an I/O exception when reading or writing data, an operator immediately notifies APC so that failure recovery can be quickly initiated.

F. Enabling Service Personalization

Not only does ICEBERG provide service mobility support to allow mobile users access services across network and device changes, it also stresses the concept of having the person instead of the device as the communication endpoint. This level of personal mobility is possible by creating a single identity for an individual providing a level of indirection to the desired endpoint for communication. A service thus can be transparently accessed by the user regardless of his endpoint communication device. Furthermore, services are customized using the client's preference specification depending on the user's current activities. The *Personal Activity Coordinator (PAC)* service [6] keeps detailed account of the current ongoing activity of the user to make services properly customized according to user's location and activity. The APC service queries the database of user's preference and constructs the appropriate data path.

G. Localization of Services

Depending on end user's current location, services can be *localized* by incorporating useful local information. For instance, a user sitting in a traffic jam moving slowly will automatically receive updates on alternative routes to his destinations as part of the service data. A student going into a Computer Science building automatically gets information about ongoing seminars. This is possible because APC is constantly keeping track of user's current location and getting updates through PAC about user's current activity and customizes the service output accordingly. Thus, services can become more context- and location-aware.

II. IMPLEMENTATIONS

In this section we describe the overall automated process of constructing a data path of operators – service instances.

A. Path Construction Process

To construct a path, the Network Service Provider for the service whose content is to be composed sent a request to the APC along with the information pertaining to the endpoints of the required path, any specific operators that must be included in the path, the optimization metric, and an acceptable range of costs for the path. Both the metric and the cost are application specific and can be one of the following: data latency, data throughput, and output data characteristics (e.g., audio/video quality, image resolution). This information is needed to construct an optimal path.

The path construction process consists of four steps. As shown in Figure 4, it is an iterative process of continuous feedback and optimization.

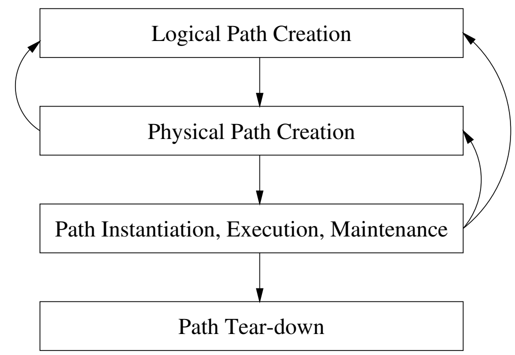

Fig. 4. **Path construction process:** Path execution is an iterative process of optimization. The APC service guarantees the availability and fault-tolerance of a constructed path by rebuilding its physical or logical path when components fail.

Step 1: Logical Path Creation: A **logical path** consists of an ordered sequence of operators joined by connectors. During the logical path creation, the APC searches through the XML descriptions of the operators available to find valid sequences that could perform the computation requested by the Application Service Provider. The result is a list of possible operator sequences ordered by decreasing cost on the user's input parameters for optimization (e.g., latency, data throughput, voice quality). The search is performed using shortest path search on the graph of operator space. Optimization criteria are application specific. One goal of the APC is to adapt to application requirements to optimize resource utilization.

Note that since some operators may be commutative (e.g., image format transcoders), the space of all possible logical paths can be huge given a large number of services. Hence, as a heuristic, only a small number of logical paths are generated initially. As indicated in Figure 4, additional logical paths can be produced as needed if the physical paths for the first set of selections are not optimal (i.e. cannot satisfy user's specified QoS metric or do not have acceptable performance). Thus, the tradeoff for better response time does not compromise the degree of optimization for quality of service.

Step 2: Physical Path Creation: A **physical path** is a logical path, along with a choice of actual nodes (i.e. physical machines) on which to run the operators. Nodes for long-lived operators are chosen from the known service instances' locations that provide the desired functionality based on application dependent criteria such as response time, data throughput, and image resolution. Nodes for dynamic operators are selected according to the computational capabilities, the cost of using that node in the path, and various other criteria. Some of these operator placement decisions include operator computational requirement, software/hardware requirement, output/input properties (e.g., data location, data volume, delay-sensitivity, degradation properties), network characteristics (e.g., bandwidth, delay, packet loss rate, and jitter characteristics). The APC constructs a physical path from a logical path by finding the lowest cost nodes that meet the requestor's requirements.

During physical path creation, optimization operators such as FEC and compression operators are inserted for performance enhancement. FEC operators are added between a wireless link to reduce packet loss rate. Compression and decompression operators are added between links with heavy data throughput to avoid overloading the network.

Step 3: Path Instantiation, Execution, Maintenance, and Querying: Once the physical path has been determined, APC starts any required dynamic operators and sets up appropriate connectors between the various operators. After all the nodes in the path are set up, the data flow is started. In addition, a **control path** (described in Section I-E) is established between the operator nodes and APC. It is used for both reporting of error conditions and performance statistics.

During the lifetime of the path, APC actively monitors the operator nodes to make sure that they are functional. Any operator can also report problems to APC about its neighboring operators, so that the path can be repaired when necessary. The control path also plays an important role in enabling operator repair, deletion, and insertion. It is used for exception handling, controlling parameters of path components, monitoring and analyzing path performance. Therefore, it needs to be highly robust and unaffected by the data path's failures. A control path can however overlap with the data path: each path operator can have a handle to its two neighboring operators. APC thus monitors the performance of the path and reroutes the data path if new conditions make the original path suboptimal by going back to the physical or logical path creation stages.

Step 4: Path Tear-Down: When a path is no longer needed, the user informs APC. The APC service then stops the data flow, removes connectors, shuts down any dynamic operators, and frees other relevant resources. As a performance optimization, APC may cache the logical and physical information of commonly used paths for reuse at a later time.

B. Operator Functional Classification

To automate the logical path construction process, APC needs to be aware of all supported operators. It is thus useful to have a meaningful categorization of operators in their functionality. The functional classification is included in the operator XML description. We classify operators into four categories: data format translation operators (e.g., GIF to JPEG converter), protocol conversion operators (e.g., security protocol conversion), content transformation operators (e.g., English to Chinese translator), and optimization operators (e.g., FEC operators). A wide-area service discovery service (e.g., [7]) keeps track of available operators for APC to create data paths.

III. Supported Applications

We have implemented these applications using APC: accessing Ninja Jukebox service [8] (MP3 format) using a cell-phone, accessing email through Vat (an audio tool) [9], communication between a PSTN phone and a GSM cell-phone, and voice-enabled interactive room control [10]. We have also explored mobility issues for supporting video streaming to wireless mobile clients. Most of the above applications are developed for the ICEBERG project [1] with the main goal to enable any-to-any communication independent of end devices. The functionality of path allows seamless integration of any new device into the communication infrastructure [11]. It only requires the addition of an operator that converts between a supported format to the new device's data format. Below we describe some of the supported applications in detail.

Our experience shows that service composition is greatly simplified by APC and the QoS of the resulting composed entity is guaranteed through the path search process. Moreover, we find the classification of operators based on the properties in Section II-B quite helpful during operator and path creation.

A. Listening to Jukebox MP3 Songs using Cell-phone

Ninja Jukebox is a distributed, collaborative music repository that delivers digital music in MP3 format to Internet clients in real-time. A GSM cell-phone user interested in using the service can take advantage of APC's data transformation functionality to convert the music into the right data format. Furthermore, the data path established also hides the network jitter and bandwidth fluctuations by buffering and downsampling the data.

This data path is built from existing Unix utility tools – *mpg123* and *sox* programs, and a GSM lossy speech compression codec [12]. This demonstrates that APC can easily integrate legacy code as operators to construct data paths.

B. Universal Inbox

Universal Inbox is an infrastructure service in ICEBERG that provides customizable redirection of incoming communication based on user preference profiles as well as user's end devices. The inbox is universal because it accepts all types of communication, e.g., voice mail, paging message, email, news feeds from the Web. Depending on user's current end device, the incoming message is automatically transformed to the proper format before delivery. For example, an HTML-email message goes through a speech synthesizer and a PCM encoder before reaching a user on a PSTN phone. For Universal Inbox, APC is the key to extensibility and service portability.

C. Interactive Voice Room Control

In this era of ubiquitous computing, it becomes important to control various appliances in smart spaces using various modes, e.g., speech, text, gesture, etc. There is such a smart space in our lab consists of various audio and visual appliances to be controlled over the network. Using APC service, one can control A/V equipment (e.g., move cameras, turn on lights, program VCR) using a variety of input devices. Paths are automatically constructed from input source to the room control application and vice versa. For instance, speech input is first converted to PCM audio, then speech-to-text conversion is performed on the output, which goes through natural language processing to the text format. The text subsequently is changed into commands accepted by the room application. Responses from the application goes through the inverse transformation: first to text, then PCM speech, and finally GSM audio if the end user is using a cell-phone.

D. Real-time Streaming Video for Wireless Mobile Clients

To demonstrate mobility support we have developed a video streaming service for mobile wireless clients through real-time transcoding and dynamic data stream redirection. As a future extension, we plan to support real-time video conferencing between wired and wireless clients using a variety of end-devices. Depending on the access network bandwidth and client's end device capability, APC automatically generates the proper sequence of transformation operators and content adaptors to generate desired data format at the proper data rate and quality. Furthermore, the constructed data path is intelligent enough to redirect data when user roams.

IV. Evaluation

To demonstrate the scalability of APC service, here we present some end-to-end performance measurements for one path application (accessing a MP3 streaming Jukebox service using a cellphone). The measurements are performed on a local area cluster of 400MHz Pentium-II machines each with 256MB of main memory and 512KB of processor cache with gigabit ethernet connection.

Table IV shows good performance of a single APC service instance for a path consisting of 4 operators on 2 nodes with 200 paths continuously being created and torn down in the background. The performance shown is acceptable because the response time for a path creation is less than 500ms. Users typically do not care about how long it takes for the service session to terminate. In this case, it takes less than 300ms. Recovery takes slightly longer (i.e. 400ms); however, if buffering is used, the user can hardly notice any gap in the output audio. The scalability of path is also reasonable – 16 paths per machine, 15 creations per second. This means that to handle 1000 service creations per second, we need 67 machines, while 200,000 service sessions are in progress.

In the context of two-way telephone calls, statistics [13] show that during busy hours, the average call arrival rate $R = 2.8\ calls/hour/user \times N$ (N is the number of users in the system), with the call duration $t = 2.6\ minutes$. From our measurements, we know that the rate of path creation is $15\ paths/sec$ with 32 paths running in the background. The call arrival rate a two-node APC can handle is therefore given by $32/(2.8 * 60) = .19\ call/sec$. Thus, the system can handle $N = .19/(2.8\ calls/hour) = 244$. Therefore, a two-node APC service can easily handle over 200 users for this type of transcoding operators: sound conversion operations and GSM to PCM codecs.

TABLE I

Performance of a single APC service instance for operating on 4-operator-paths on 2 nodes. There are 200 paths continuously being created and destroyed in the background on other nodes.

Logical and physical path creation time:	264ms
Path instantiation time:	215ms
Path teardown time:	289ms
Path recovery from one failed operator:	402ms
Data throughput:	64kbps
Path construction latency:	479ms
Path scalability:	16 concurrent paths
Single-Node APC throughput	15 creations/sec

V. Related Work

We now examine related efforts focusing on seamlessly interconnecting Internet services and resource-constrained devices. The main distinction is that our architecture provides fault-tolerant, wide-area, and scalable composition automation of both legacy and novel services with dynamic optimization of resource utilization. The resources we consider include computational, memory, and network resources. The optimization criteria (e.g., latency, jitter, data throughput) are either defined by the service authors or deduced from the type of the service. Our emphasis is on reusing existing services and enabling a quick and easy way to obtain new service functionality from existing ones rather than building a very complex and difficult-to-evolve service accommodating a fixed set of protocols and devices. Existing work addresses only specific aspects of the problem space. Additionally, one of our contributions is to provide a well-defined framework for Internet service composition and a taxonomy of computation paradigms to provide differentiated quality of service (QoS).

Our work is influenced by flexible middleware systems supporting distributed computing across heterogeneous resources. For example, Corba [14] provides platform-independent, object-based network communication. DCOM [15] is an equivalent of Corba for the Windows platform. However, neither system directly supports optimal placement of computations. Jini [16] is a Java-centric view exploiting bytecode mobility to deliver stub code implementing a private communication protocol between client and service. Nevertheless, it is mainly designed for use on a much smaller scale than wide area, e.g., workgroup. HP's e-Speak [17] does target for wide area operation but does not address service scalability and fault-tolerance; nor does it provide access from simple devices that are incapable of running java-based communication protocol.

Our work is heavily influenced by projects (e.g., [18], [19], [20], [21], [22], [23], [24]) that transcode to adapt service content to better suit impoverished small devices. However, these approaches are vertically integrated. They do not use composition as a way to easily extend system functionality. Furthermore, quite a number of works have tried to develop a single interface for large classes of devices (e.g., WAP protocol [25]). The success of our work, however, does not depend on the adoption of a single standard. We provide bridging of multiple standards by providing translational elements across them by designing an extensible architecture to adapt to future standards.

On the surface, our design goals are similar to active networks [26], which allow code injection into network routers to deploy new network protocols, implement traffic shaping, and perform packet filtering. Our architecture allows placing computation within the network with the goal of performance and resource usage optimization. The distinction is that for active networks, data processing occurs at transport or packet level rather than using application semantics. In our work, operators are application-oriented and process application data rather than router-level packets.

The idea of path or composition, similar to UNIX pipeline-like chaining of different commands, existed in many previous works. For example, Scout [27] uses the path as an explicit communication-oriented abstraction in operating system design. In Scout, path facilitates OS specialization by enabling configurability and exposing global context that optimization techniques can exploit. Path also assists resource allocation and management by being a single unit of scheduling entity. Infospheres project [28], as another example, focused on building compositional systems from interacting components. MPLS (Multiprotocol Label Switching) [29] defines the abstraction of a label-switched path to provide QoS in IP networks. Here, we extend this idea of composition to wide-area, independent Internet services. The extension includes automatic path formulation as well as runtime path maintenance.

Our work can be considered as an extension to the TACC programming model [30] with additional design of wide area service placement, continuous resource optimization through feedback, and generalized load balancing. TACC model provides composition of stateless data transformation and content aggregation with uniform caching of data. The composition model is static and inflexible. We are exploring a completely automated composition model and programmable compilation of composition chain. Furthermore, services considered in our framework are quite general, including continuous latency-sensitive stream services such as live audio and video as well as support for mobile wireless clients. These are not addressed by TACC.

VI. Conclusions and Future Work

We have presented the design, implementation, and evaluation of an infrastructure service – the Automatic Path Creation service, the key enabling component to achieve service and personal mobility, as well as service portability and customization for integrating heterogeneous networks and devices in ICEBERG architecture.

Such a service composition platform automates compositions of both legacy and new Internet services across the wide area. Automation is enabled through a strong type system and the encoding of service attributes through flexible XML descriptions. By providing features of fault-tolerance, scalability, and optimized adaptive resource utilization, we allow service authors to focus on the specific content of their services rather than how the service will be accessed by different devices and how it will interoperate with other services. We achieve fault-tolerance through redundant control paths responsible for fast fault-recovery. Scalability is leveraged by using a cluster computing platform. Optimized resource utilization and differentiated QoS are obtained through the iterative path construction process with continuous feedback and a clear specification of the user's optimization criteria. Moreover, we identify two key abstractions (i.e., operator and connector) useful for putting together services in a tinkertoy-like fashion and for service component reuse. In the future, we plan to explore support for more diverse clients (e.g., IPAQ, Palm Pilot), and integration of more existing Web services. We would also like to deploy our implementation in wide area to stress test scalability and fault-tolerance features of our prototype.

VII. Acknowledgment

I thank Professor Randy Katz, my advisor for his guidance and encouragement since I joined the ICEBERG research group. I also thank Professor Brewer and Professor Culler for their insightful comments and ideas. I owe a great deal to members of the Ninja and ICEBERG project for their input.

References

[1] H. J. Wang, B. Raman, C-N Chuah, R. Biswas, R. Gummadi, B. Hohlt, X. Hong, E. Kiciman, Z. Mao, J. S. Shih, L. Subramanian, B. Y. Zhao, A. D. Joseph, , and . R. H. Katz, "Iceberg: An internet-core network architecture for integrated communications," *IEEE Personal Communications*, August 2000.

[2] Steven D. Gribble, Matt Welsh, Rob von Behren, Eric A. Brewer, David Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. Joseph, R.H. Katz, Z.M. Mao, S. Ross, and B. Zhao, " The Ninja Architecture for Robust Internet-Scale Systems and Services," *To appear in a Special Issue of Computer Networks on Pervasive Computing*, 2000.

[3] D. Clark and D. Tennenhouse, "Architectural Consideration for a New Generation of Protocols," in *Proceedings of SIGCOMM '90*, Philadelphia, PA, 1990.

[4] Real.com, `http://service.real.com/help/library/blueprints/8codecs/producer8codecs%.html`, *Working with RealProducer 8 codecs*, June 28, 2000.

[5] T. E. Anderson, D. E. Culler, and D. Patterson, "A Case for NOW (Network of Workstations)," *IEEE Micro*, February 1995.

[6] Xia Hong, "Personal acitivity coordinator: A coordination layer for independent services," M.S. thesis, U.C. Berkeley, December 1999.

[7] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, and R. Katz, "An Architecture for a Secure Service Discovery Service," in *Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99)*, August 1999.

[8] Ian Goldberg, Steven D. Gribble, David Wagner, and Eric A. Brewer, "The Ninja Jukebox," in *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems, Boulder, CO.*, October 1999.

[9] `http://www-nrg.ee.lbl.gov/vat/`, *VAT Mbone Audio Conferencing Software*.

[10] A. D. Joseph, B. Hohlt, R. H. Katz, and E. Kiciman, "System support for multimodal information access and device control," Workshop on Mobile Computing Systems and Applications (WMCSA), 1999.

[11] Bhaskaran Raman, Randy H. Katz, and Anthony D. Joseph, "Universal Inbox: Providing Extensible Personal Mobility and Service Mobility in an Integrated Communication Network," WMCSA 2000.

[12] Jutta Degener, "Gsm 06.10 lossy speech compression," `http://kbs.cs.tu-berlin.de/~jutta/toast.html`.

[13] 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.

[14] The Common Object Request Broker Architecture, *The Object Management Group (OMG)*, `http://www.corba.org`.

[15] G. Eddon and H. Eddon, *Inside Distribued COM*, Microsoft Press, Redmond, WA.

[16] Sun Microsystems, *Jini Connection Technology*, `http://www.sun.com/jini/`.

[17] Hewlett Packard Inc., *eSpeak: The Universal Language of E-Serices*, `http://www.e-speak.net/`.

[18] Charles Brooks ad Murray S. Mazer, Scott Meeks, and Jim Miller, "Application-Specific Proxy Servers as HTTP Stream Transducers," in *Proceedings of the Fourth International World Wide Web Conference*, December 1995.

[19] M. Liljeberg and et al., "Enhanced Services for World Wide Web in Mobile WAN Environment," Tech. Rep. C-1996-28, University of Helsinki CS Department, April 1996.

[20] Y. Sato, "DeleGate Server," `http://wall.etl.go.jp/delegate/`, March 1994.

[21] M.A. Schickler, M.S. Mazer, and C. Brooks, "Pan-Browser Support for Annotations and Other Meta-Information on the World Wide Web," in *Fifth International World Wide Web Conference (WWW-5)*, May 1996.

[22] B. Schilit and T. Bickmore, "Digestor: Device-Independent Access to the World Wide Web," in *Sixth International World Wide Web Conference (WWW-6)*, Santa Clara, CA, April 1997.

[23] Bruce Zenel and Dan Duchamp, "A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment," in *Thrid Annual ACM/IEEE Conference on Mobile Computing and Networking (Mobicom'97)*, New York, NY, 1997, ACM.

[24] Ka-Ping Yee, "Shoduoka Mediator Service," `http://www.shoduoka.com`, 1995.

[25] WAP Forum, *Wireless Application Protocol (WAP) Forum*, `http://www.wapforum.org`.

[26] David J. Wetherall, John Guttag, and David L. Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols," in *Proceedings of IEEE OPENARCH'98*, San Francisco, CA, April 1998.

[27] D. Mosberger and L. Peterson, "Making Paths Explicit in the Scout Operating System," in *Proceeds of OSDI'96*, 1996.

[28] Caltech Infospheres group, "Caltech Infospheres Project," Joint W3C/OMG Workshop on Distributed Objects and Mobile Code, June 1996.

[29] Vijay Srinivasan George Swallow, "Multiprotocol label switch-

ing (mpls)," http://www.ietf.org/html.charters/mpls-charter.html, Nov. 17th, 2000.

[30] Armando Fox, Steven D. Gribble, Yatin Chawathe, and Eric A. Brewer, "Extensible Cluster-Based Scalable Network Services," in *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP-16), St. Malo, France, October 1997.*, 1997.