Achieving Service Portability using Self-adapative Data Paths

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

Abstract— **There is a growing demand for service access through heterogeneous devices attached to diverse networks. For ease of deployment, it is crucial to provide application-level support for transparent roaming and ubiquitous service access. In this paper we discuss how to develop such a service infrastructure integrating a variety of telephony and data services spanning diverse access networks reaching heterogeneous end devices. We describe our techniques for achieving goals of transparent network- and device-independent service access, as well as scalable and fault-tolerant access to composed service entities across the wide area using *self-adaptive data paths*. We evaluate our implementation through applications that require adaptation to end devices and resource variations. The applications include Universal In-box, Interactive Voice Room Control, MP3-Jukebox access using a cell-phone, and real-time video delivery to wireless clients.**

I. Service customization using paths

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 Personal Digital Assistant (PDA) 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 conferencing call, she is reminded again by her PDA 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 PDA with 802.11 network connectivity. Since 802.11 may have a much lower bandwidth, the image size and quality is automatically reduced. The video input is received by the camera attached to the PDA. 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 our service architecture is the middleware service – **Automatic Path Creation (APC) service** [1]. The APC provides data transformation for handling format mismatches and also provide adaptation to resource variations. The 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 **data path**. It consists of a directed graph of **Operators**, which 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. We achieve automatic composition of operators by mandating services to be strongly-typed. 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

The APC allows services to be accessed transparently from any device and any network using data format transformation operators. Given a data format mismatch, the 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.

The APC service is completely transparent to end-users and *application service providers* (e.g., Yahoo). It 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 the APC to build a data path.

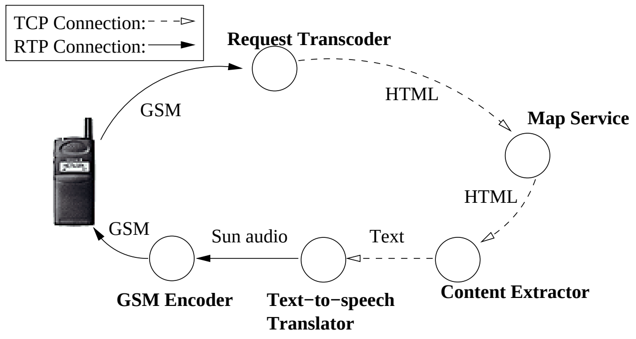

Fig. 1. Example path use scenario: Illustration of two paths created for getting map directions using a cell-phone from a map service. Each circle denotes an operator which can be either a long-lived service instance (e.g., Map Service) or a dynamically created one (e.g., GSM Encoder).

Mismatches between an end-user and service occur because the user lacks the necessary software or hardware, or may have resources that are too limited like 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 the APC. Similarly, thin PDA clients with limited display, computational, memory, or power capabilities can conveniently use any existing legacy services without specialized services. 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, the APC also provides personal mobility support needed by mobile clients. A roaming client switching between wired and wireless access may change its IP address. The APC enables it to continue the ongoing service session with minimal interruption. The data path provides a service session redirection proxy 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 for retransmission to provide the illusion of a continuous session.

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

C. *Dynamic adaptation to resource variations*

Frequently end users may experience degraded performance of service due to dynamic changes in network conditions, e.g., sudden bandwidth drop due to network congestion. There is a need for applications to adapt to dynamic changes in available resources to optimize user-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. RealVideo uses a single codec to encode data for several bandwidths. During runtime, the audio and video streams dynamically adapt to changes in bandwidths [2]. In this case, the 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 instances of the same codec 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, the APC is responsible for monitoring the resource changes and providing feedback to the applications to enable dynamic adaptation. the APC continuously monitors resource changes and the quality of output data to detect the need for dynamic path optimization.
- **Application-independent adaptation:** If there is a lack of knowledge of the underlying application implementation, the 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 are used. Given the path requestor's optimization criteria, the 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. 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 the elimination of coupling and blocking effects within the service composition chain and thereby greatly increasing 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 deploy 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.

E. Achieving fault-tolerance for composed services

The APC protects 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 faults. 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, the APC service instances on different clusters coordinate to build a wide area path shown in Figure 2.

The 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.

- **APC monitoring**

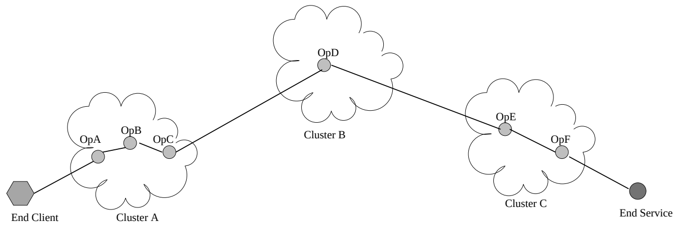

Fig. 2. An example wide area path: Each circle denotes an operator. Each connecting line between the operators indicates a connector. The path is requested by the end service.

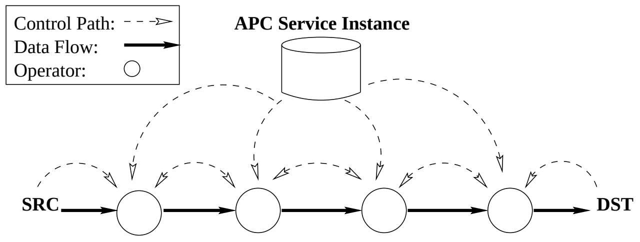

Fig. 3. Redundant control paths: To ensure a fast and robust fault-recovery model, multiple control paths are built into the system to guarantee their robustness.

The APC periodically sends heartbeat messages to each operator within the path to ensure they are available. Upon timeout, the APC assumes either process or node failure and attempts to restart the operator. First it tries to reuse the existing node if it is not overloaded. If that fails, the 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 any I/O exception when reading or writing data, an operator immediately notifies the APC so that failure recovery can be quickly initiated.

F. Enabling service personalization

Not only does our service architecture 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 [3] 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 the 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. APC can be either a stand-alone infrastructure service or a part of the Network Service Provider's functionality. The major difference between a self-adaptive data path and a simple concatenation of a series gateways is that the data path constructed by APC is customized for the application and dynamically adapts to changing resources. This adaptation is achieved seamlessly by APC using passive monitoring.

A. Path construction process

To construct a path, the Network Service Provider for the service whose content is to be composed sends a request to the APC at a well-known IP address 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.

Step 1: Logical path creation: A **logical path** consists of a directed graph of operators. During the logical path creation, APC searches through the XML descriptions of the available operators to find valid sequences that could

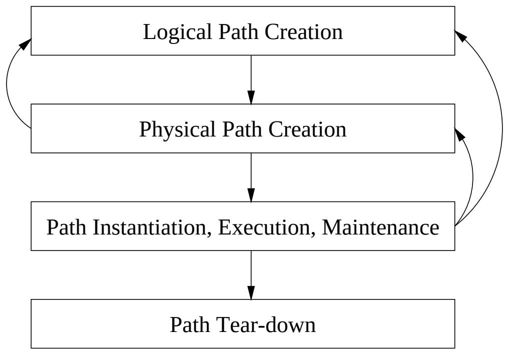

Fig. 4. Path construction process is an iterative process of optimization. The APC service guarantees the availability and fault-tolerance of a constructed path by rebuilding its physical or logical path when components fail.

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 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 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). 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 operators with large data throughput to avoid overloading the network.

Step 3: Path instantiation, execution, maintenance: 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 available. 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 notifies APC, which 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.

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 keeps track of available operators for APC to create data paths.

III. Supported Applications

We have implemented these applications using APC: accessing a music MP3 jukebox service using a cell-phone, accessing email through Vat (an audio tool) [4], communication between a PSTN phone and a GSM cell-phone, and voice-enabled interactive room control [5]. We have also explored mobility issues for supporting video streaming to wireless mobile clients. The functionality of path allows seamless integration of any new device into our service infrastructure. It only requires the addition of operators that convert between supported formats to the data format of the new device. Below we describe some of these applications in detail.

Our experience shows that service composition is greatly simplified by APC and the QoS of the resulting composed entity is significantly improved 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

The music jukebox [6] 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 [7]. This demonstrates that APC can easily integrate legacy code as operators to construct data paths.

B. Universal inbox

Universal Inbox [8] is an infrastructure service providing 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 the Universal Inbox, APC is its 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, 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 go 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. 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 and changes access network.

IV. Evaluation

To demonstrate the scalability of the 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.

In the context of two-way telephone calls, statistics [9] 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.

Our work is influenced by flexible middleware systems supporting distributed computing across heterogeneous resources. For example, Corba [10] provides platform-independent, object-based network communication. DCOM [11] is an equivalent of Corba for the Windows platform. However, neither system directly supports optimal placement of computations. Jini [12] 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.

Our work is heavily influenced by projects (e.g., [13]) 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. 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.

The idea of path or composition, similar to UNIX pipeline-like chaining of different commands, existed in many previous works. For example, Scout [14] 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. 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 [15] 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.

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 achieved by using a cluster computing platform and event-driven programming style. 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. In the future, we plan to explore support for more diverse clients (e.g., IPAQ) 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

We thank Professor Eric Brewer and Professor David Culler for their insightful comments and ideas. We also thank members of the Ninja and ICEBERG project for their input.

References

[1] 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.

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

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

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

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

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

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

[13] 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.

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

[15] 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.