ADAPTIVE RESOURCE MANAGEMENT OF A VIRTUAL CALL CENTER USING A PEER-TO-PEER APPROACH

Munir Cochinwala, Namon Jackson, Hyong Shim, and Eric Sigman
Applied Research, Telcordia Technologies Inc
445 South Street
Morristown, New Jersey 07960, USA
{munir, namon, hyongsop, erics}@research.telcordia.com

Abstract: As the number and diversity of end user environments increase, services should be able to dynamically adapt to available resources in a given environment. In this paper, we present the concepts of migratory services and peer-to-peer connections as the means of facilitating adaptive service and resource management in distributed and heterogeneous environments. Our approach has been realized using object-oriented principles in Adaptive Communicating Applications Platform (ACAP). The architectural design and implementation of a real-life high-level service, Virtual Call Center (VCC), are used to illustrate issues in adaptive service and management issues and discuss in detail our approach in ACAP.

Key words: Adaptive and Distributed Service and Resource Management, Peer-to-Peer Application, Service Platform, and Virtual Call Center

1. INTRODUCTION

With the increasing availability of network connectivity and network-enabled devices at work, at home, and on the road, users will require that services be adaptive to their environments and devices. Already, some popular services, such as email and instant messaging, are available on PCs, PDAs, and cell phones. However, the existing practice of creating multiple versions of the same service for execution on different devices is inefficient, unproductive, and wasteful of development and management resources. Ideally, services should be built once and

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35674-7_66

G. Goldszmidt et al. (eds.), *Integrated Network Management VIII*

be able to dynamically adapt to available resources and capabilities in diverse environments. Thus a mechanism is needed that allows service providers to effectively deliver and manage services in different environments

In this paper, we describe a peer-to-peer approach for managing end user environments and resources in a scalable and flexible manner. An environment mainly refers to a computing/communication platform from which the user accesses their services, and a resource refers to a specific hardware device or application tool used in a service. In our approach, each environment manages its own resources and is aware of their capabilities, and services negotiate with a given environment for required resources at runtime. An environment could be a home network, an ISP or a provider network. The approach is peer-to-peer in that once a service is activated in an environment, service endpoints communicate with the environment for resource requests. Subsequently, service endpoints share data and resources directly among themselves. Any accounting and state update information at the end of a service session is relayed to the service provider on an as-needed basis.

Since each environment is self-managed, the load increase on service providers needed to support new environments is incremental compared to a centralized approach, in which service providers manage all the environments and resources themselves. This helps increase the scalability of services and allows for rapid introduction and management of new services Our approach is also adaptive in that resource availability can be locally maintained and environment management can be tailored to the specific characteristics and requirements of individual environments.

Our approach has been incorporated in a prototype platform, called Adaptive Communicating Applications Platform (ACAP) [1]. ACAP takes an object-oriented approach to managing services and resources. Specifically, in ACAP, services specify resource requirements, and environments provide resources that match the requirements--multiple resources may match specific requirements. Services dynamically execute in diverse environments by adapting to available resources by migrating and requesting available resources from environments at runtime. Migratory services are facilitated by use of migratory objects.

ACAP is a working prototype and has been used to develop a number of applications to show the viability of our approach to service and resource management. In this paper, we present one such application, Virtual Call Center (VCC), to illustrate service and resource management issues in distributed and heterogeneous environments. We also describe in detail how ACAP is used for the development of the VCC application. Note that ACAP is a general-purpose platform that can be used for any application that requires integrated service and resource management in distributed environments.

The rest of the paper is organized as follows. Section 2 describes VCC and its service and resource management issues. Section 3 provides an overview of ACAP. Section 4 describe and discuss in detail the use of ACAP for VCC. Section 5 discusses related work of ACAP with emphasis on industry efforts on service management. Section 6 describes future work and concludes the paper.

2. VIRTUAL CALL CENTER: A MOTIVATING EXAMPLE

A virtual call center (VCC) consists of operators who are geographically distributed, and on duty at different times. When a customer calls, VCC's call processing/scheduling engine determines an operator and routes the call to the operator. In this paper, we show the call processing/scheduling engine as a centralized dispatcher. Dispatching functionality can be distributed across environments or implemented in distributed databases such as in 800 call routing [10]. Either of these mechanisms could easily be incorporated into our approach.

One critical issue in increasing the effectiveness of the VCC (or a regular call center) is state transfer. A typical user experience with a call center is that the customer often has to repeat relevant information while being transferred from one operator to another. This problem is even more prevalent with the advent of the Web, where users often perform a lot of "work" before calling the VCC, e.g., putting potential purchases into online shopping carts, filling out electronic order forms, and registering with merchant sites.

VCC can allow operators to effectively determine what callers are trying to do without requiring further, duplicate information by seamlessly transferring caller registration, history, state, and other information. This dramatically enhances the quality of user experience and increases the likelihood of turning users into customers. Furthermore, a centralized approach of having server components always manage state transfer and update transactions would not scale to a large number of callers and any server failure may disrupt the services of all callers, which would significantly degrade the effectiveness of the overall system. Thus, a distributed approach is needed that efficiently allows for scalable, robust, and fast state transfer. In this paper, we describe a peer-to-peer approach, in which server components "get out of the way" once an operator is connected to a caller, and the operator is responsible for updating the state information of the caller at the VCC once the call is complete or on an as-needed basis.

Another issue involves the means by which VCC operators and callers communicate and exchange information. Different callers may have different preferences for how to communicate with the operator. Some may prefer text-based instant messages, while others may prefer more interactive, real-time communications. In the latter case, the caller preference may range from a telephone/cell phone to a VoIP call to a multimedia call that includes voice, video, and documents. In order to accommodate diverse caller preferences, the communication mechanism used by the VCC should be adaptive to end user environments and available resources. At the same time, it is unreasonable to expect that the VCC would/could pre-provision all the computing/network environments of distributed operators to meet all caller requirements; doing so would be prohibitively expensive. Therefore, an efficient mechanism is needed to dynamically discover and adapt to available resources and their capabilities in diverse operator environments. We use the concept of migratory services to address this issue. Specifically, calls are represented as services that can dynamically move to operator and caller environments and execute using locally available or preferred resources.

3. ACAP

Adaptive Communicating Applications Platform (ACAP) is an object-oriented service platform that provides support for developing and managing high-level services in a distributed and peer-to-peer manner. In this section, we provide a high-level overview of its basic constructs: services and environments. In Section 4, we describe in detail how the VCC system is built using these constructs.

The main objective of ACAP is to provide support for adaptive services. An adaptive service can change the way it operates, depending on available resources in environments. To illustrate, a two-party voice call service can be made adaptive by facilitating the parties involved to use the communication devices of their choice, e.g., a regular phone for the caller and a PC-based phone for the called party. In ACAP, the concept of a migratory service is the main means by which adaptive services are realized. The basic mode of operation in ACAP is that a service endpoint first moves to an environment, negotiates for the available resources in the environment, and then executes using the negotiated resources. This way, services adapt to environments in a distributed manner, in which much processing occurs at or near endpoints. This reduces processing overhead on servers, which, in turn, helps increase the scalability of services. To facilitate migratory services, ACAP uses migratory objects. Specifically, a service is modeled as a collection of service objects that can move based on user request and system resource availability. A service object implements service logic, maintains service state, and specifies a resource requirement for the service. A service is a special service object that also functions as a container of other service objects. A service may be contained in other service containers, thus forming a hierarchy of services. Henceforth, the terms, service and service object, are used interchangeably, unless a distinction needs to be made.

Services and their service objects always maintain their containment relationships, even when they are at different locations. When a service moves, only those objects that are co-located with the service also move. In ACAP, we use remote references to represent inter-object relationships. A remote reference is similar to an ordinary object reference, except that (1) when a remote reference is serialized, it does not serialize the referee, and (2) when a serialized remote reference is recreated, it points to the original referee, even if the recreation is on a different machine. Remote references are used for passing objects in (possibly remote) method invocations and also allow migratory service objects to maintain their links to other service objects.

In ACAP, environments are modeled in terms of available resources and their capabilities. For example, a PSTN phone is modeled as a resource capable of audio communication with a certain bandwidth requirement. An environment is modeled as a resource, which in turn, is a collection of other resource objects. In this way, the entire environment, including the network, can be modeled hierarchically. It also allows us to efficiently compute and manage the state of an environment and by induction, the state of any service or system.

Each environment can independently administer its own resources. Policies may be used for security and/or enforcing user preferences. This allows us to efficiently

and independently represent and introduce new resources into the system without central administration and provisioning. Rapid introduction of new resources requires that new devices and services are deployable by multiple third party providers.

When a service enters an environment, ACAP binds each service object with a resource object that can meet its requirements. For example, a call service object may require resources for inputting and outputting audio, for which ACAP may bind a resource object that represents a regular phone, an IP phone or a PC audio system to the call service object. The decision as to which resource objects are bound to service objects depends on resource availability in a given environment and user preferences.

In our current implementation, resource requirements are hard requirements. If no resource is available, then the service will not be able to execute. We already have the notion of service adaptation to equivalent resources such as different types of phones. We are designing mechanisms for services to adapt to degraded resources such as insufficient bandwidth for video streaming. We can easily adapt to degraded resources using service specific logic such as the designer of the service replacing streaming with periodic still images. We are currently exploring generalized mechanisms for resource degradation (non-availability is the worst kind of degradation).

4. ACAP AND VCC

VCC makes use of ACAP support for adaptive services to provide scalable and adaptive call center services. Specifically, VCC models a call as an adaptive service that can dynamically move to the environment of the VCC operator who gets assigned to the call. By executing the received call, the operator can retrieve registration, history, current state, and any other relevant information of the caller and establish a peer-to-peer communication channel with the caller. In fact, a call is a container object that has other objects with caller information and VCC call service logic. In this section, we describe and discuss in detail use of ACAP in the development of the VCC application.

4.1 Architecture

Figure 1 graphically shows an architectural overview of the VCC in relation to ACAP. VCC DISPATCHER is the central administrative hub for the system. It maintains a database of customer registration information, operator information (e.g., their schedules, locations, and state), and other system resources. It also handles all the incoming calls and assigns them to appropriate and available operators, the process of which is described shortly.

In Figure 1, VCC OPERATOR is an ACAP-based tool that implements the VCC application logic for VCC operators. It provides a notification service that alerts the operator. Furthermore, it can interact with the tools and resources that are available

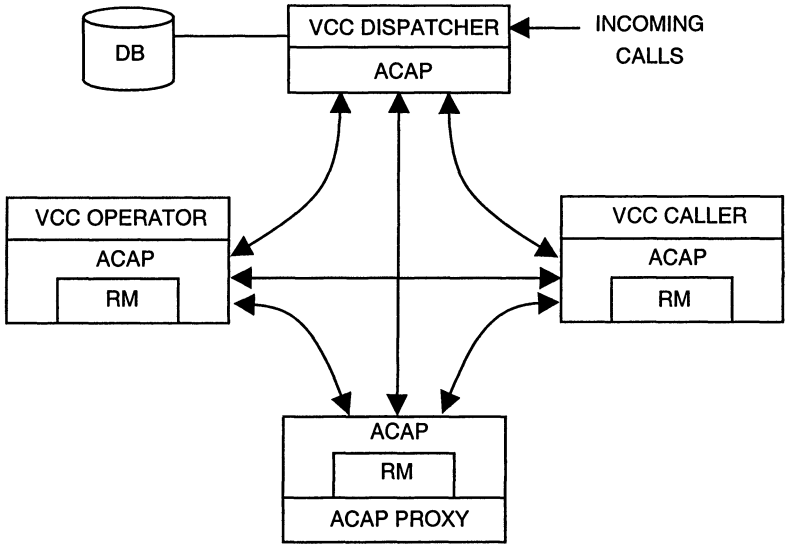

Figure 1. Architectural Overview of the VCC Using ACAP

in the operator's environment via the ACAP Resource Manager (RM), which is described shortly.

VCC CALLER is an ACAP-based tool that enables the caller to interface with the assigned operator. It establishes a peer-to-peer communication channel with the VCC OPERATOR of the operator and creates collaboration/communication sessions between the caller and operator on an as-needed basis. Currently, it is assumed that the VCC CALLER is pre-installed and configured in callers' environments, say, as part of creating a subscription with a service provider. However, it could also be provided as an applet that can dynamically be downloaded via a Web browser and installed and configured for the caller's environment when a call is made.

Callers may use regular or cell phones to contact the VCC. In such a case, the VCC OPERATOR connects to the ACAP PROXY to establish a voice channel between the caller's phone and the operator's device for voice communication. ACAP PROXY is a logical entity that may include PSTN/VoIP gateways to allow use of SIP phones or VoIP application tools. ACAP PROXY keeps track of the capacity and current usage state of its PSTN/VoIP gateways. VCC may have multiple instances of ACAP PROXY, in which case the VCC DISPATCHER may dynamically determine a particular instance to be used based on the caller phone number and the location of the assigned operator.

In Figure 1, the flow is from a caller to the dispatcher, which migrates the call to the appropriate operator so that the operator can communicate directly with the caller. Resources are allocated to the call once it migrates to the environment of the assigned operator. In the next sub-sections, details of interactions among various components are described.

4.2 ACAP Resource Manager (RM)

In ACAP, Resource Manager (RM) manages devices, application tools, and other resources that are available in a given environment. Typically, RM runs in the same environment as the one whose resources it manages (as is the case for the VCC system). However, if infeasible to do so, RM can also run at a remote location.

Resources are associated with a hierarchical, ACAP-defined type system. Figure 2 shows a partial resource type hierarchy and example resource instances for a typical VCC operator environment based on a networked desktop computer. Each resource type has a set of attributes that describe capabilities and other properties. For example, the resource.comm.audio.ip type represents resources for VoIP communication and has properties, in and out, which represent audio input and output resources/devices respectively. The same resource type hierarchy is used by ACAP services to specify required resources.

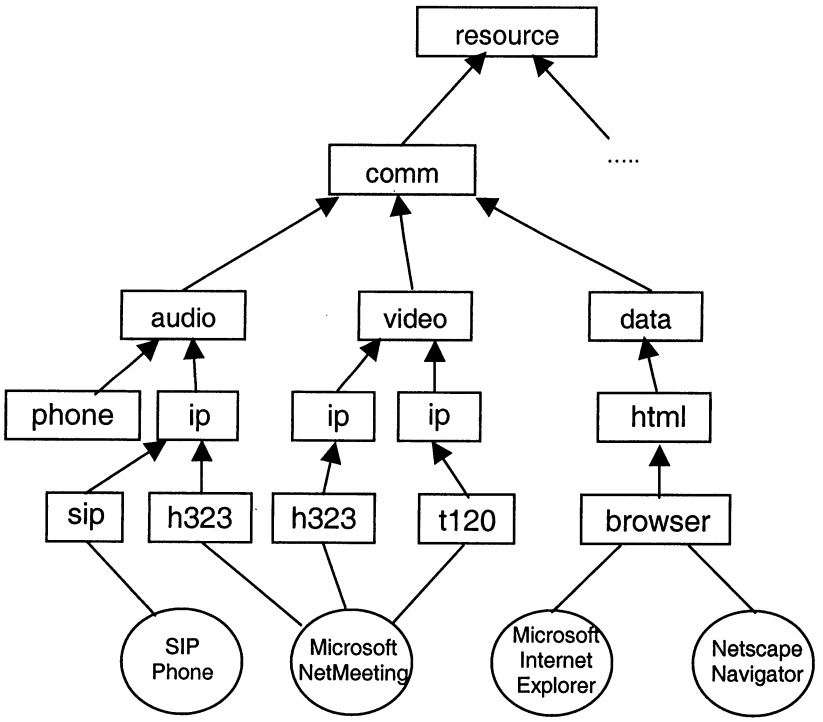

Figure 2. A Partial Resource Type Hierarchy and Example Resource Instances for a VCC operator environment. A resource type is a rectangular box, and a resource instance is a circular box.

Part of the VCC OPERATOR/CALLER installation and configuration process is to construct a resource hierarchy for an environment, most of which can automatically be deduced from the local (file type ←→ tool) association settings. Multiple instances may exist for a given resource type, e.g., resource.comm.audio.ip and resource.comm.data.html.browser in Figure 2, in which case the operator/caller is prompted to specify preferences.

When a service enters an environment, each of its contained service objects makes a request to the RM for a required resource. If the requested resource is available, the RM returns the corresponding resource object. The resource object is a proxy for the real device or application tool and provides an interface through which the service object can control, communicate, and manage the device or application tool. Communication between a resource object and the corresponding device or application tool is device or application-dependent. For example, the resource object for Microsoft NetMeeting may use a COM interface of this tool that allows other applications to control it programmatically. This way, existing tools and devices can easily be integrated with an ACAP environment, which in turn helps reduce the "learning curve" both for VCC operators and callers.

4.3 Handling Incoming Calls

Upon receiving a call, VCC DISPATCHER performs the following tasks:

- Retrieve the caller history (based on caller id), current state (e.g., URL history), address, and preferred method of communication of the caller.
- Create a CALL service object, which is a topmost container that represents the current call and contains USER, CONTEXT, and COMM service objects. The USER object stores any registration information of the caller, the CONTEXT object any received state information, and the COMM object the caller's current location and preferred methods of communication in terms of resource types. Each of these service objects has an application-specific interface.
- Create and store a copy of the CALL object for record keeping and persistence. This is the primary copy and holds the true state of the call if the information is lost in transit.
- Select an operator and put the CALL object in the QUEUE of the selected operator. QUEUE represents a communication channel between the VCC DISPATCHER and VCC OPERATOR of the selected operator. The exact operator scheduling and selection criteria may be policy-dependent and are beyond the scope of this paper.

When the CALL object arrives at the operator site, ACAP notifies the operator-end of the QUEUE, which, in turn, notifies the VCC OPERATOR. VCC OPERATOR starts or activates the new CALL and alerts the operator of its arrival. When the operator wishes to communicate with the caller, the VCC OPERATOR makes a request to the COMM service object, which first asks the ACAP RM for a communication resource of the same type as that of the caller's top preference. When the RM returns a resource object (see Section 4.2), the COMM creates two

ACAP Endpoint objects. Each Endpoint is a service object that represents an end point of a "call" of a specific type and is to be bound to a resource object for a device or application tool used for the call. Binding an Endpoint to a resource object may involve starting an application tool and retrieving its address information, e.g., the IP address and port of the host computer, on which the application tool is running. In ACAP, this address information is called the resource address of the Endpoint. Subsequently, the COMM asks the RM to bind the operator Endpoint to the returned resource object and then sends the other Endpoint to the caller.

On the caller's side, the VCC CALLER receives and starts the caller Endpoint object, at which point the Endpoint asks the RM in the caller's environment for a communication resource of its type. When the RM returns a resource object, the Endpoint object binds to it and alerts its counterpart at the operator's site of its resource address information. Subsequently, the operator's Endpoint makes a request to its resource object to initiate a communication session, passing it the resource address of the caller Endpoint. In turn, the resource object instructs its application tool or device to "call" the specified destination.

Figure 3 graphically illustrates the architecture of an example communication session between the VCC operator and caller, the set-up process of which is just described. This architecture applies to all types of communication in ACAP. The operator and caller Endpoints may continue to exchange information even after a communication session has been established. For example, in a collaborative Web Browsing session, the operator Endpoint may send to the caller Endpoint the URLs of the Web pages that the caller's Web browser should display and vice versa. For "multimedia" sessions, ACAP allows multiple pairs of operator ←→ caller Endpoints to exist and operate at the same time.

Note in Figure 3 that the resource objects do not have to run on the same host as their application tools or devices. This design is critical in increasing ACAP's ability to adapt to the preferred or available resources. To illustrate, consider a case where the operator and caller wish to have a voice communication, and one or both of the parties wish to use a regular phone. Here, the resource object cannot control the phone directly. Instead, it interfaces with a gateway to the phone network, which most likely is located at a remote site, and control the phone via this gateway. Furthermore, the operator/caller may be on a device with limited resources or capability, e.g., a phone. ACAP can still accommodate such a case by running the resource and Endpoint objects on a proxy host. Also note in Figure 3 that the COMM object on the operator site still contains the caller Endpoint object, even after it has moved to the caller's site. This way, the COMM object can still maintain context across contained objects in presence of the mobility.

When the call is complete, the VCC OPERATOR requests that the CALL release all the allocated resources. Furthermore, it notifies the VCC DISPATCHER of the call completion and sends it any state updates by sending the CALL and its contained objects back. Subsequently, the VCC DISPATCHER updates its database with any updated information from the received CALL. It may be possible that the connection between the VCC OPERATOR and DISPATCHER may fail while the operator and caller communicate. In such a case, the VCC OPERATOR locally stores the CALL and its contained objects until the connection is restored. The state

of the call is contained within the object and can be updated in the future, even in batch mode, if desired.

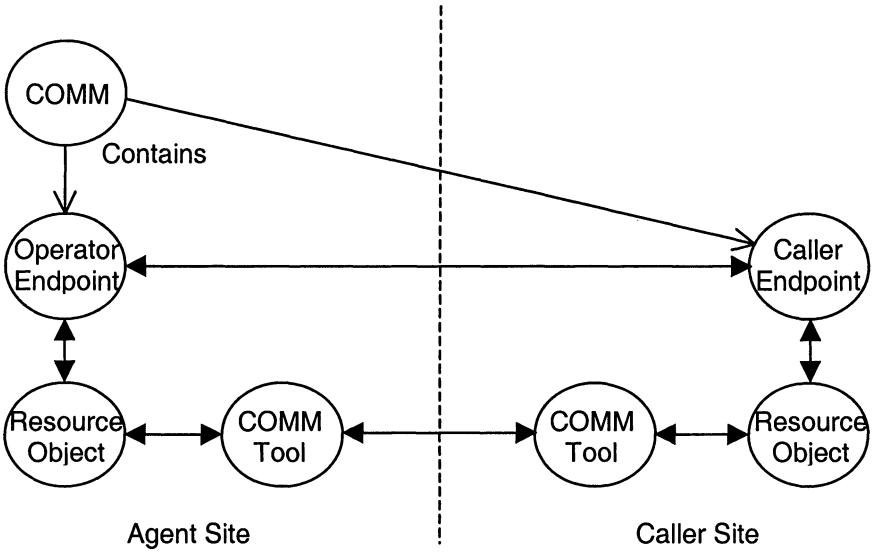

Figure 3. Example Architecture for Operator and Caller Communication

5. RELATED WORK

To our knowledge, ACAP is the first adaptive services platform that uses the concept of migratory services and peer-to-peer connections as the main means of developing and managing high-level services. However, the idea of adapting services to available resources in end user environments has been receiving an increasing amount of attention. For example, Open Services Gateway Initiative (OSGi) [2] is an industry consortium that specifies an object-oriented framework for remotely delivering and managing services. The OSGi framework provides a common life-cycle management service and allows services from different vendors not only to co-exist in the same environment but also dynamically discover and make use of each other. In addition, it allows a variety of end user devices to be represented and integrated with other services.

Note that the OSGi framework does not specify the mechanics of service operation, whereas ACAP specifies a model of service operation using migratory objects. This makes ACAP and OSGi complimentary to each other. For example, ACAP can use the service/device management facilities of the OSGi framework in implementing its Virtual Call Center (VCC) application as follows. ACAP Resource Manager can be implemented as an OSGi service bundle. This bundle

can be downloaded and installed on the OSGi framework of a call center operator, at which the Resource Manager, once activated, can discover communication and other resources of the operator's environment. The ACAP call service can also be implemented as an OSGi service bundle, which can dynamically be downloaded and activated and discover the local Resource Manager. Integration with the OSGi framework is part of the future work.

Telecommunications Information Networking Architecture (TINA) [3] is an example of high-level service deployment and management platform that takes a client-server approach. Mainly developed from the viewpoint of service providers, TINA specifies a set of architectural principles and object-oriented information models for next generation multimedia telephony services. TINA takes a client-server view of how services should be provided in that every aspect of managing a service involves server components "running in the network." For example, in TINA, creating end-to-end communication channels involves a Communication Session Manager (CSM), which is mainly responsible for collecting and distributing the address information of end user devices to be used.

In contrast, ACAP is inherently peer-to-peer. In ACAP, the basic model of communication is to create and send peer service objects to communicating endpoints. Once in place, these objects connect to each other directly. This way, much processing of the application logic of a service can take place directly in end user environments, and server components mostly perform administrative tasks such as billing and subscriber management. Thus we argue that ACAP provides better support for scalable service environments than TINA.

Object mobility has received a lot of attention in the literature. One of the main applications of object mobility has been in localizing access to distributed objects in order to increase system performance. The main idea is to move an object to where it is needed, which makes accessing the object not much more expensive than local method calls. In order to automate the decision as to when and where to move objects, a variety of system- or language-level support is provided. For example, Kan [4] extends the Java language to include Kan-specific keywords and constructs for asynchronous method calls and transactions and keeps track of read/write access patterns on Kan objects. When a Kan object is write-accessed one thread at a time, it migrates. Otherwise, the object is replicated to where read requests are issued, and write requests are sent to its home site. StratOSphere [5] allows migratory objects to adapt to the available resources of new sites by enabling them to adopt the local implementations of their methods. The Aleph toolkit system [6] facilitates efficient location of migratory objects via its Arrow distributed directory protocol. Migratory objects, in the form of mobile operators, have also been used in the area of active messaging, e.g. [7].

In ACAP, migratory objects do not contain any "intelligence." Rather, they are used in designing and managing migratory and adaptive services in a distributed environment. Specifically, they are mainly used as the means of transporting application code and data to end user environments and establishing peer-to-peer connections between communicating endpoints.

6. FUTURE WORK AND CONCLUSION

ACAP has effectively been used for rapid development of VCC and other advanced services, and its capabilities have successfully been used in both internal and external demonstrations. However, a number of important issues still remain. One such issue is security. Specifically, ACAP dynamically moves among communicating endpoints services objects that may contain executable code. This raises the issue of how to trust the authenticity and integrity of received service objects. On the other hand, service object owners should have control over who has access to their objects and where they move. In the former case, a PKI-based approach may be used to sign and verify the integrity of migratory service objects. In the latter case, we have a capability-based [8] approach, which allows owners to keep track of and revoke capabilities even when their objects are at remote locations. Fully addressing these and other security issues in ACAP is part of future work.

Another area of interest is to apply ACAP approach to management of network resources. Specifically, our model of individual environments managing their own resources can be extended to include network-level resources. That is, when a service migrates and asks for resources, the RM can grant or deny requests based on both current network conditions and locally available resources. This way, network operators/service providers can have fine-grained control over access and utilization of their resources. Providers can also implement resource usage-based billing.

Integrating network resource management with service creation and management is an important area for providers. This will allow providers to manage and prioritize limited manpower and network resources to revenue generating services. We are actively working in this area developing a service model that allows service definition across network and service layers.

In summary, ACAP applies object-oriented principles to managing services and environments. In ACAP, services are specified in terms of required resources, and environments in terms of available resources. ACAP facilitates services to adapt to diverse environments by allowing them to dynamically migrate and discover available resources in a given environment. Migratory services are facilitated by use of migratory objects in ACAP. In this paper, we have described in detail a Virtual Call Center (VCC) system to illustrate adaptive resource management requirements for high-level services. Furthermore, we have shown our approach to addressing these issues by describing how support for migratory services and peer-to-peer connections in ACAP is used in its architecture and implementation.

REFERENCES

[1] Alberi, J., Cochinwala, M., Cohen, E., Jackson, N., Pucci, M., and Sigman, E., "An Object-Based Framework for Communication Services," IEEE GlobeCom 2000, Workshop on Service Portability and Virtual Customer Environments (SerP-2000), December 2000.

[2] "OSGi's Service Platform Release 2" available at http://www.osgi.org.

[3] Abarca, C., et al., "Service Architecture," TINA-C Deliverable available at http://www.tinac.com/specifications/documents/sa50-main.pdf.

[4] James, J. and Singh, A.K., "Design of the Kan distributed object system," Concurrency: Practice and Experience 12(8): 755-797, 2000.

[5] Wu, D., Agrawal, D., and Abbadi, A.E., "Mobility and Extensibility in the StratOSphere Framework," Distributed and Parallel Databases 7(3): 289-317, 1999.

[6] Demmer, M., and Herlihy, M.P., "The Arrow Directory Protocol," Proc. of 12th International Symposium on Distributed Computing, September, 1998.

[7] Okino, C. and Cybenko, G., "Modeling and Analysis of Active Messages in Volatile Networks," Proc. of the 37th Allerton Conference on Communications, Control and Computing, Monticello, IL, 1999.

[8] Landwehr, C.E., "Formal Models for Computer Security," ACM Computing Surveys, Vol 13, No. 3, September 1981.

[9] http://www.recursionsw.com/products/voyager/voyager.asp.

[10] Cochinwala, M., "Database Performance for Next Generation Telecommunications," Proceedings of International Conference on Data Engineering 2001: 295-298.