Using Objects for Next Generation Communication Services

Munir Cochinwala

Telcordia Technologies, 445 South Street, Morristown, NJ 07960 USA
munir@research.telcordia.com

Abstract. The integration of the telephone network and the internet enables convergence of voice and data services. The explosion of information appliances also provides new service opportunities. Object-oriented systems show great promise for building new classes of services that dynamically adapt to changing networks and appliances. This paper describes how we use migratory objects to build new classes of services. The focus is on telecommunication services where voice is integrated with other types of media/data.

1 Introduction

Recent dramatic improvement in data networking technologies have dramatically changed the landscape for traditional providers of telecommunication services. Data networks are now able to provide adequate quality telephone service, both in access networks (e.g. IP over cable) and in backbone networks (e.g. various forms of IP telephony over ATM). Telecommunication providers see a significant threat to their revenue. The presence of IP telephony providers and open competition in the long distance market has resulted in price reductions ranging from from 20% in the U.S. to 70% in Germany.

However, significant new revenue opportunities are available to telecom providers who can expand into new markets. The market size is significant. Consumer spending on communication services such as telephony, cable, internet access and premium services such as Video on Demand are predicted to be around $100 billion in 2001 with telephony services accounting for $48 billion.

The underlying technological trend is the explosive growth of data networks. The percentage of network bandwidth occupied by voice traffic is predicted to decline from 40% today to less than 5% in 2005. The optimal strategy for telecommunication may be handle voice as much as possible like data.

This new integrated network is termed the Next Generation Network (NGN). A NGN is a network where all kinds of information (voice, fax, video, data) are transmitted uniformly using packet-based transport and switching media.

The challenge for telecom providers is to deploy new, innovative services that integrate the traditional ease of use and reliability of the phone network with the vast new potential of the data network and the web.

Using the flexibility of NGN, a new integrated multi-service communications framework can be created. Using this framework telecommunication companies

Elisa Bertino (Ed.): ECOOP 2000, LNCS 1850, pp. 388–393, 2000.

can bundle new services with traditional services and reverse the trend of losing revenues to internet, cable and entertainment companies.

At Telcordia we have developed a framework for NGN services based on objects and object migration. Our goal is to develop a framework where new services can be rapidly developed and deployed. We believe that our framework meets goals of seamless integration of the phone network and data network incorporating the best that each network has to offer. The new framework requires a significant change in the infrastructure used for call (communication) set-up.

2 Object Framework and Applications

Our goal is to provide a framework where new services can be developed and deployed rapidly. Our focus is more than simple telephone calls, it is communicating applications. New application should be able to use the new services as easily as they would make a telephone call.

We have designed a basic set of building blocks that can be assembled to create new services. A communicating service begins with a pair of coupled objects which are the communication endpoints. For a typical 2-party communication, one endpoint is sent to the destination party. A third party call set-up would require sending both sides to different destinations. Local resource managers are responsible for allocation of resources such as audio devices for normal telephone calls or screen real estate, camera, etc for more complex interactions. The communicating service can also be created by cloning an existing communicating service.

To place an ordinary telephone call, a user creates two endpoints, sends one to the intended recipient, attaches the other to a local device (e.g., a phone). When the peer arrives on the recipient's desktop, it will attempt to get resources and give an alert (e.g., ring the phone). If the originating user does not immediately attach his endpoint to a device, we are left with a symmetric situation resembling a "hot line": each side has an endpoint for the call, but no resources (or signaling) will take place until one side picks up. This can be used to implement a variety of interesting services (e.g., an anonymous dating service).

The endpoint objects can be migrated to different devices or destinations. We use object mobility as the basis for migration. One might ask why we bother using mobility at all, rather than just using standard distributed object technology (e.g., CORBA). From our perspective, object mobility is simply distributed objects, along with (1) transparent message forwarding, and (2) automatic code distribution. The second capability is not necessarily critical in our system (where we emphasize object composition over new code creation). For our purpose, the main advantage of automatic message forwarding is that it enables greater decoupling between clients and servers.

A user may wish wish to integrate a number of service capabilities into a single handler. For example, he might have a number of objects able to display media of different types, and one of these may be chosen for an incoming media stream. Or he might have access to a number of different service providers

offering a common capability (e.g., bit transport), and he wants to choose the one offering the best rate. Such management policies can be embedded in specialized objects, which we call "service containers". These containers behave like ordinary desktop containers (i.e., managed objects can be dragged in and out), and provide coherent presentation and management of their contained services.

We have focused on building framework objects that can be used to implement complex business function. For example, object containment can be used to implement complex services such as bundling voice and data. Object containment can also be used to maintain application context when items are passed among participants. We have used these concepts to implement a tele-medicine application.

Using our framework, the tele-medicine application can be designed so that even non-technical people like doctors and nurses can make effective use of the flexibility of services. The doctor can make a phone call to another doctor, patient or pharmacist. During the conversation, any one of the participants can choose to share or include a data item and use the phone call as a channel to share or broadcast data to the other participants. Private communication between any subset of the parties can also be easily accommodated by establishing a private channel or by using private data in objects. All of these functions can be accomplished within the phone call or any other communication. The phone call is just another application and to the end-user or the developer, it is a convenient channel that is currently configured to handle audio. The channel can dynamically adapt to other media types.

In our prototype, we demonstrate the following features:

- Information passed between patient, doctor and pharmacist is encapsulated in objects
- Prescription and patient history are bundled with voice communication and migrated across the parties
- Database access and transaction context of patient history can be shared and protected
- The infrastructure supports customized handling of media types based on user preferences. These preferences may be to send data to one network while the voice goes to another network.

3 Services Model

We now compare our object-based framework to traditional implementation of telephony services. We focus on the differences in call set-up and call models.

Current telephony systems are based on centralized creation and deployment of relatively static services, carefully designed to interoperate safely. Conversely, on the Internet, anybody can create and deploy a new service, but these services cannot interoperate effectively.

In traditional telephony, there is a sharp division between services and sessions. Services are usually written in a programming language, carefully integrated

with other system components, with severe security requirements, while a session is created by a user (or a service), and has a relatively short duration/value. An important thrust of our work is to eliminate the distinction between these two types of entities.

In telephony, it is common to use very different protocols for the originating and terminating endpoints of a call. This approach greatly complicates the design of many services (e.g., third-party call setup). Instead, we try to make use of call models that are completely symmetric.

We have used object and object migration as the fundamental method of communication. We now present a comparison of the our approach compared with the traditional approach for telephony.

- In traditional telephony call setup is based on a universal addressing scheme. That is, the initiator decides on the address of the responder (possibly after some universal number translation phase, like 800 number translation, or some multilevel lookup, as in call forwarding or internet DNS), and requests a connection to that endpoint.
  In our framework, all application addressing is to objects; the client never sees where the message is actually routed. Objects can migrate while the system is running; this migration is transparent to clients/communication peers.
- Traditionally, there is a previously agreed-upon protocol regulating communication between the initiator and responder; this protocol regulates how parties can join, suspend, and terminate a call, what sort of busy/failure signals can be conveyed, etc. This makes changes to the call model extremely difficult; evolutions to the call model are very rare, and must always maintain backward compatibility.
  In our system, the call initiator chooses the complete protocol to be used for the communication. Of course, by choosing a protocol where the terminating endpoint is expected to provide a standard terminating half call model, he can get the effect of the current call model. However, he can also choose a completely different call model, requesting direct interaction with devices at the terminating endpoint.
- Telephone calls require initiating and terminating sides to have very different roles. This is because conventional calls are relatively short lived, during which the initiator (but not necessarily the terminator) is assumed to be participating.
  Conversely, our call models are generally symmetric. This is because the calls are possibly long-lived, with either or both parties temporarily or permanently suspending participation, without the call being torn down (from the logical standpoint - an implementation may choose to tear down the actual circuit for efficiency reasons).
- Audio is the media of choice in telephony and is normally determined at call setup time (perhaps through a process of negotiation between the two parties). In contrast, in our calls, the initiator (or, more commonly, the call model) obtains a handle to a virtual environment of devices (resources)

belonging to the terminator. The call model can allocate and/or deallocate resources, on either side of the call, at any time.
Resource allocation in our framework is carried out through resource managers. A resource manager is an object that can be asked to deliver other objects, given a description of the object desired (typically its type). When a call (or any other communication object, such as a document or an advertisement) wants to make use of resources (such as stable storage, a speaker, or an alerting device like a ringer), it gets suitable objects from the resource manager. (Because devices like phones can themselves be viewed as being composed of many subdevices, the resources are themselves resource managers.)

– Security is maintained by associating actions with principals (typically representing people or organizations), and annotating sensitive resources (like files and devices) with lists of principals (called access control lists). When an action tries to access a resource, the system checks that the principal responsible for the action is on the list of allowed accessors; if the principal is not on the list, the access fails. However, this makes it difficult for one party to hand access off to another, which is an essential paradigm in our system.
Our system uses what is called capability-based access control: a handle to an object itself guarantees access. Since handles can be freely copied and sent to other principals, this allows one participant to hand access off to another participant.
The usual problem with capability-based access control is what is called the confinement problem: once you give away a capability, you can never get it back. We solve this problem by making extensive use of a proxy discipline. When a resource manager wants to delegate use of a resource to some party that it does not fully trust, it creates another object of the same type that forwards requests on to the original resource. However, this new resource also has a "kill" switch, accessible only to the resource manager, that can be used to dissociate the proxy object from the original. (The proxy is also designed so that when it gives out a subresource, these subresources are also given out as proxies, which can be killed when the original proxy is killed.) This allows the resource manager to revoke previously issued resource grants at the cost of a level of indirection). (It can also preprogram the proxies to kill themselves, either after some time interval or after a certain number of accesses.)

4 Conclusion

We have a prototype built using the framework. We have incorporated multiple devices in our prototype ranging from Java Rings, MP3 players, IP phones, and gateways to traditional telephone networks. The infrastructure allows setting up communication channels, attaching resources (e.g audio) to communication channels and migration of live communication across devices and users. Furthermore, a complex object (voice and data) can be combined or split and sent

to different destination (users) on different networks. The immediate next steps are to incorporate our framework into NGN products and deploy the products.

Deployment of the framework and applications on a world-wide basis poses tremendous challenges. Challenges from traditional system design are the scalability and reliability of the new applications. Security is a major concern that we have left unexplored. Definition of a security policy and implementation of security for objects that can appear in a user's environment are avenues for exploration.

We have object mobility as the basis for communication. The implementation and definition of the object mobility platform is an area of further research. A simple question is how much of an object to move when the top level object moves. Moving the complete object may have performance impacts while moving a partial object leaves room for failure due to network partitions or object repository failure.

Finally, telecom and network providers have to design schemes for managing applications that are long-lived and may require high bandwidth. Monitoring and managing these applications may require cooperation between application objects and network management software for efficient allocation of resources.