Supporting Both Client-Server and Peer-to-Peer Models in a Framework of a Distributed Object Management System

Tonghyun Lee, Jieun Park, Seok Hwan Yoon, Pyeong Jung Kim, and Bum Ju Shin

Visual Lang. Sec, Electronics and Telecommunication Research Institute, 161 Kajong-Dong Yousong-Ku, Taean, 305-345, Korea(ROK)

Abstract. Application models of distributed applications are various so that a model appropriate for an application can be inappropriate for other applications. For example, client-server model is the most generally used model for distributed applications but peer-to-peer model is more appropriate for groupware applications. Although supporting several different application model in a framework can give flexibility to design and implement distributed applications, the difference of communication type and communication entity management should be handled in consistent and uniform way. In our distributed object system, Distributed Object Management System (DOMS), both client-server and peer-to-peer models are supported in a well combined model and the application models are supported in distributed objects level so that to design a distributed application can be flexibly done with high level abstractions and the combined model with distributed access controls can support other communication types like player-viewer model. In this paper, the combined model to support both model in a framework and the variance of the basic models will be discussed with the design and implementation issues.

1 Introduction

Within the high speed network environment, the necessity of distributed applications as distributed DB, audio-video conference, multi-user games, and etc is being increased. Requirement of those applications are various and their application model can be different from each other. Even the well-known client-server model is appropriate in many cases, it can not satisfy all needs. Thus if several necessary models can be supported in a framework of systems or tools, development of distributed applications can be done in flexible and natural ways. However, the main problems to support different models in a framework are to combine their different communication style and communication entity management in a coherent and uniform way.

In the most frequently used distributed application model, client-server, client application initiates the communication between the client and server while the
resources are maintained in the server applications. Access requests of clients can be occurred any time and if one is occurred, the server processes the request with the resources but each request is processed independently from each other. So a request from a client is not affected by other requests from other clients. A distributed DB can be a typical example application using the client-server model. Many distributed object system support the client-server model in the distributed object level by using proxy and remote objects [6].

Another important distributed application model is peer-to-peer model [8]. In this model each peer application can initiate communication any time. Thus a peer application can send requests to another peer application while it processes the requests from the other peer application. The model is typically used for the replicated applications as groupware applications. Because each replicated application has the same replicated resources and functionality, they should communicate each other in the same manner. This model can be implemented by using replicated objects [2].

In DOMS, the two models are supported and combined by providing proxy, remote, and replicated objects. DOMS is a run-time object system supporting those types of objects and it provides developing tools and API for general distributed applications including groupware applications.

In this paper the way to combine two different models in distributed object level and the concepts of each object type are discussed with their design and implementation issues.

2 Object Types for Application Models

The basic distributed object model of DOMS is C++ object model extended for concurrence and persistence [1-3]. They can be classified as proxy, remote and replicated objects in view of distributed objects. The relation between a proxy object and a remote one is shown in Figure 1. The server application maintains remote objects and it processes the access requests from client applications and returns the processing results. The access requests for a remote object are delegated to its proxy object in the client application. Proxy objects do not keep any data but interface the access of the related remote object so that any update requests from a client are reflected in the remote object in a server but any other update of the remote object invoked by the server or other clients is not notified to the client.

![Fig. 1. Communication between a proxy and a remote object](image)