SHARED OBJECT MANAGEMENT SYSTEM

FOR PEER-TO-PEER APLLICATIONS

Sehoon LEE
Department of Information Systems
College of Computing Sciences
New Jersey Institute of Technology, Newark, NJ, 07102, USA
seihoon_lee@hotmail.com

and

Ungu KANG
Department of Newmedia
GachonGil College, Inchon, 405-701, Korea
ugkang@gcgc.ac.kr

ABSTRACT

In this paper, we proposed a shared object model, an component-based P2P(Peer-to-Peer) model, which complies with an open architecture, and manages resources afforded through a new component called SOC(Shared Object Class) in order to solve problems attendant upon a P2P model.

Also, it is so designed as to enable it to be extensible in a variety of ways under a P2P environment through a distributable pluggable function. The SOC comes up with a new model featuring in particular efficient distribution of computing power and resources and automated management, thus overcoming weaknesses encountered in building an enterprise information system on the existing P2P models.

Keywords: P2P Model, Shared Object Model, Resource Share, Distributed Environment, Peer Computing

1. INTRODUCTION

As high-speed internet connection services and high-performance personal computers develop, existing client-server computing environment has been changed into peer computing which is a distributed server environment where network effects can be maximized[1, 2, 3]. Peer computing is a P2P(Peer-to-Peer) service where peers in the P2P network share computer resources and services by direct exchange[2, 4]. Since the resources of the information equipments which can be connected through the internet and clients in both sides can share each other's resources in equal aspect, peer computing can be applied into various fields including file sharing, distributed computing, collaboration, and intelligent agents[2, 5, 6, 7]. The major goal of these P2P application in various fields is the efficient sharing of various resources. However since implementing a specific application for each resource requires enormous amount of duplicate investments in time and expenses, development system is keenly necessary. However existing development system[8, 9, 10, 11] shows some problems in dynamic scalability, usability, and shared resource management due to excessively sticking to architecture[2, 12, 13].

Therefore, in this paper, we propose a shared object management system which manages resources in the form of object in order to solve the existing P2P problems including reliability, usability, and scalability. This system introduces the new type of resource wrapper which is SOC(Shared Object Class), makes the resources into objects, and provides a method for accessing the encapsulated objects through resource specification according to the object descriptor. The system classifies the shared resources according to the types, provides standardized interface modules, isn't affected by specific operating system, protocol, and database in the P2P network environment, and provides a method which can handle various resources like network including file sharing, storage equipment, printer, offline connection service as object-based resources. Accordingly this

system can provide scalability and interoperability which are required in P2P model.

2. RELATED WORKS

While there are quite a few P2P-related research projects now under way, among those projects proposing architecture are Legion and Magi™ Open-Source Infrastructure proposed by the members of the Intel' s P2P Working Group, the ' e -speak' , an open source project now under way in HP, Project JXTA SUN[App00, Bol00, Hp00, Pro01].

Legion is an extensible, integrated architecture for secure resource sharing in distributed, heterogeneous, multi-organizational context, engineered from first principles to meet the challenges of a P2P environment[Bol00]. While Legion is a model that meets requirements under a variety of environments that include object model, security, scalability, extensibility, site autonomy, heterogeneity, NFS & Samba support and distributed events, it was not intended for the purpose of P2P environments, its key function being storage management under distributed environments. However, the possibility of its application under a P2P environment was proposed recently.

Magi are an architectural framework that explicitly addresses the coordination of e-business messaging and deployment across a range of computing platforms. It is an open-source interoperability specification consisting of complementary protocol standards, formats, and implementations. In spite of various strengths it has, Magi being an architectural framework designed to perform projects related with the processing of objects on the level of containers, it is difficult to maintain consistency relative with peers in case specifications of containers have to be altered. Also, without reference to specific models or types of resources being shared, it is realized with respect to the processing of events utilizing a Servlet engine through HTTP alone.

An open source project under development by HP, the e-speak is an architecture that enables a variety of P2P-based services to be operated through the standardized API and e-speak engine[Hp00]. A sort of service container, the e-speak engine is now getting under way, while the target of P2P services is being expanded on the basis of B2B to include applications, computing resources, business processes and the like. The e-speak features its ability to support resources aimed at P2P services in a variety of ways, and its features being a P2P model aimed at B2B. At the same time, the e-speak proposes a network object model concerning exchanges of component models of the P2P network and shared objects.

3. DESIGN OF THE SYSTEM

3.1 OVERVIEW OF SYSTEM ARCHITECTURE

The purpose of shared object management system is to provide ways of processing a variety of services (for example, ASP, etc. supporting resource sharing service) including file sharing as a concept of object-based resources without being affected by operating systems, protocols, databases, etc. under P2P environments. To this end, the system defines resources in terms of SOC (Shared Object Class), a new form of class, and provides a variety of additional functions that include reading, revising, redistributing and support for monitoring and security. At the same time, the system proposes distributive P2P service operation framework and SORBA service, a network model, and API so that it may support the SOC and the system service under diverse applications.

As shown in Figure 1, the system is composed of SOC and Object Manager intended to manage SOC, Module Container providing operation environments intended to handle service and SOC Request Manager which is a messaging sub system. The proposed model supports CORBA(Common Object Request Broker Architecture) as a middleware for inter-operability among objects, and the Java-based API in order to provide easy programming environments under diverse hardware and operation systems. For mutual communications, the RMI and RMI over IIOP are employed.

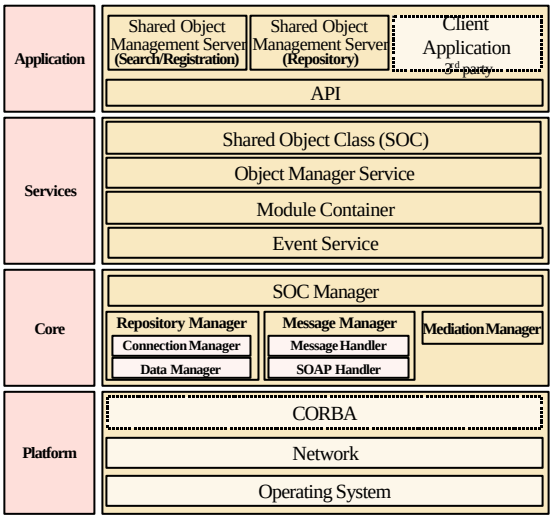

Figure 1. Hierarchical architecture of system

3.2 SHARED OBJECT RESOURCE

The resources of the system has classified to five shared resource as analysis specific properties and characteristics each other, off-line, hardware, file sharing, contents, ASP(Application Service Provider). Also the system support value added functions depend on reference and service method of module class, such as simple shared object class, variable shared object class, and composite shared object class.

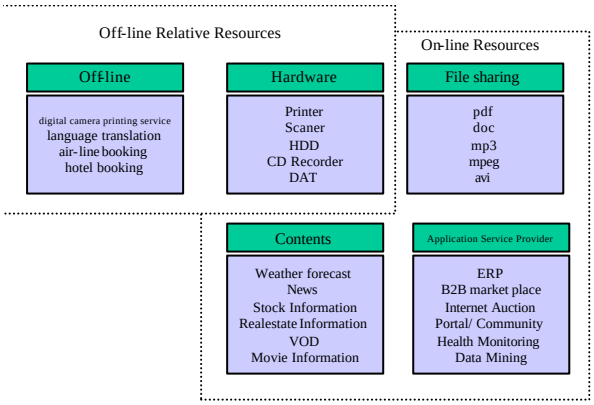

Figure 2. Categories of shared resource

A part of XML DTD(RDF) of SOC basic information is following list

```
<!ELEMENT rdf:RDF (rdf:Description)* >
<!ATTLIST rdf:RDF %rdfnsdecl; %dcnsdecl; >
<!ENTITY % dces "dc:title | dc:creator | dc:subject | dc:description
|dc:publisher | dc:contributor | dc:date | dc:type | dc:format
|dc:identifier | dc:source | dc:language | dc:relation | dc:coverage
|dc:rights" >
<!ELEMENT rdf:Description (%dces;)* >
<!ATTLIST rdf:Description about CDATA #REQUIRED>
<!ELEMENT dc:title (#PCDATA)>
<!ELEMENT dc:creator (#PCDATA)>
<!ELEMENT dc:subject (#PCDATA)>
<!ELEMENT dc:description (#PCDATA)>
<!ELEMENT dc:publisher (#PCDATA)>
<!ELEMENT dc:contributor (#PCDATA)>
<!ELEMENT dc:date (#PCDATA)>
<!ELEMENT dc:type (#PCDATA)>
<!ELEMENT dc:format (#PCDATA)>
<!ELEMENT dc:identifier (#PCDATA)>
<!ELEMENT dc:source (#PCDATA)>
<!ELEMENT dc:language (#PCDATA)>
<!ELEMENT dc:relation (#PCDATA)>
<!ELEMENT dc:coverage (#PCDATA)>
<!ELEMENT dc:rights (#PCDATA)>
```

Shared resources of these sorts should be registered as services, and the registration of service is easily realizable through the system API. Also, resources made without regard for the system can be registered and operated through the system gateway and external service interface.

The Shared Object Class (SOC), a key element of the system, is composed of resources to be shared with, pluggable functions and objective descriptor concerning the SOC. The purpose of redefining resources as a new object called SOC is to enhance reliability by preventing computing power from being lowered in order to manage resources and solve problems of resource security through distribution of functions for the management of resources after they are included in the resources. This can also enhance the worth of services by including not only simple files but also diverse functions designed for various purposes.

The functions included in the SOC could be embedded functions or it could be managed as a small object that does not exceed the size of the original file too much when the current model has been applied by designing in such a way that functions present in a remotely located object may be designated. The SOC is managed by the Object Manager, and resources that befit services as a class that can be transformed into a paralleled system can be attached to the SOC, and can be remotely located.

Accordingly, the information on the location of resources is described in terms of URI(Uniform Resource Identifier), which indicates the information on the access of objects on the Object Descriptor. The purpose of managing resources through the URI at the SOC is to operate resources in a distributive manner depending upon computing and network environments while maintaining the information on the proprietor, generator and computer.

3.3 SOC MANAGER

SOC Manager exchanges all the system messages among services as an interface to receive SOC requests from remote peers. The SOC Manager uses the Outgoing Queue to receive messages from the user. The Message Service of system brings up the Naming Resolution and Message Handler.

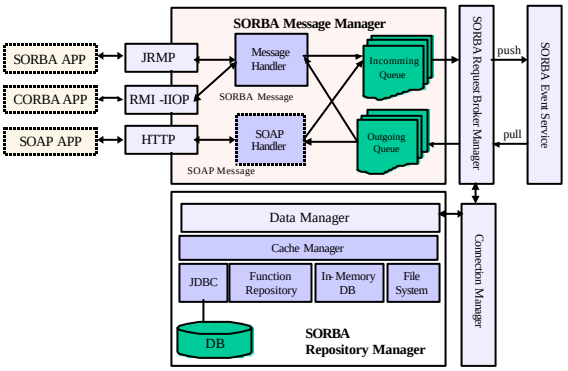

Figure 3. Configuration of SOC manager

And the Message is written down on the recipient's Incoming Queue by means of the Message Service. While

the majority of the system service handlers are located in the exterior of the system (or application), the Account Manager, Remote Resource Manager, Meta Data Manager, Incoming Queue, Function Repository, Naming Service, etc. as interior modules are located in the Module Container. Figure 3 shows basic composition of the Manager, which determines if the request from the peer is simply a reference or a duplicate of resources, and provides supports that the SOC can be readily handled at the Application through the Remote Resource Manager, Function Manager, MetaData Manager located in the Module Container. At the same time, it plays the role of registering services and providing information on the services through conjunction with the system Repository Manager.

Algorithm 1. Binding algorithm when access a object reference to remote SOC.

```
Begin Method
Monitor := get_monitorReference(monitorName);
If (monitor = not null) then begin
   nodeId := get_localHost Address ();
   monitor- >locate_node(nodeId);
      while (is_application_alive) begin
         messageType := get_message_type();
         message := get _messageReference();
      if (messageType = bind ) then
         bindingTime := get_SystemTime();
      elseif (messageType = bind_failed ) then
         monitor- >bind_failed();
      elseif (messageType = bind_succeeded ) then begin
         bindingTime := bindingTime - emTime();
         get_t atgetObject ();
end
```

3.4 SOC MANAGEMENT SERVICE

SOC Management Service take a responsibility of creation, destroy, reference, and distribution of SOC. The Service include business logics for SOC handling in applications. Figure 4 show basic composition of the management service.

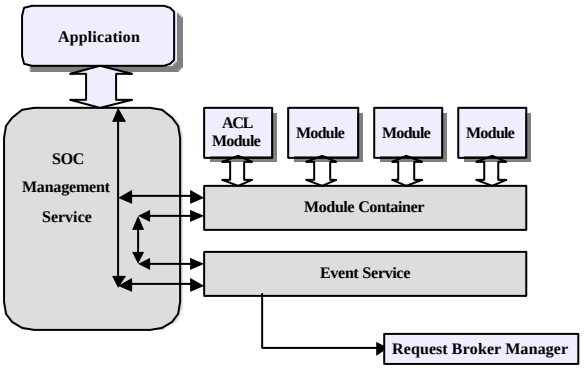

Figure 4. Configuration of SOC management service

Figure 5 show class relationship diagram of the management service for referencing and processing of SOC.

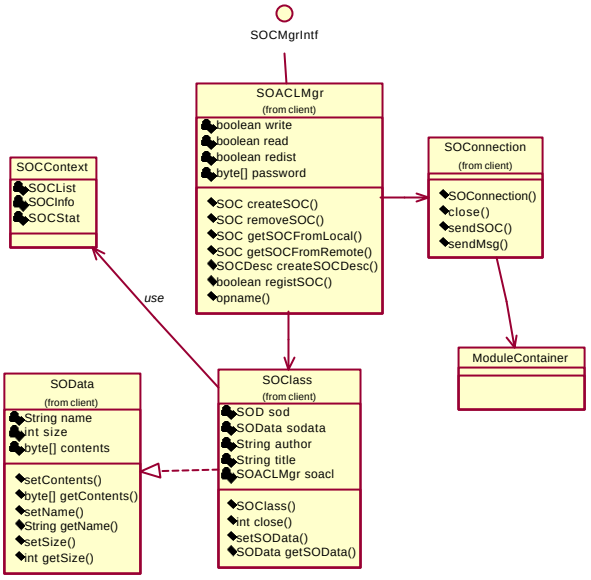

Figure 5. Class diagram of SOC management service

4. EXPERIMENTS AND EVALUATIONS

Experiments are as follows. Firstly they implement P2P applications using the system and general applications. Secondly they measure the performance changes due to applying shared object model. Finally they specify resource types in the shared object model and measure the overall efficiency due to applying shared object classes. Environment for implementation are following.

- Shared object management server : Solaris8
- Application development language : J2SE1.3

JAXP(XML Java API for XML Processing) and JAXM (Java API for XML Messaging)

4.1 EXPERIMENTS

The purpose of this experiment is to present development of P2P applications for resource sharing and to evaluate the reliability of successful requests and responses attendant upon increase in the number of queries with regard to distributed shared object resources. Accordingly, this experiment is designed in order to touch off requests and responses of differing client numbers on network, and evaluate the effects. Figure 6 and Figure 7 are sequence diagram for experiment and user interface for SOC creation each.

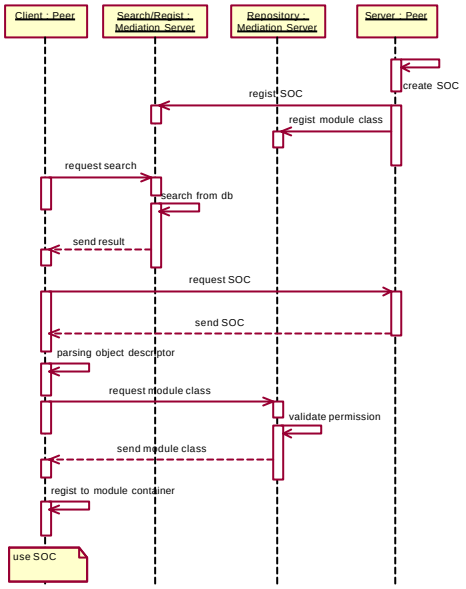

Figure 6. Sequence diagram for experiment scenario

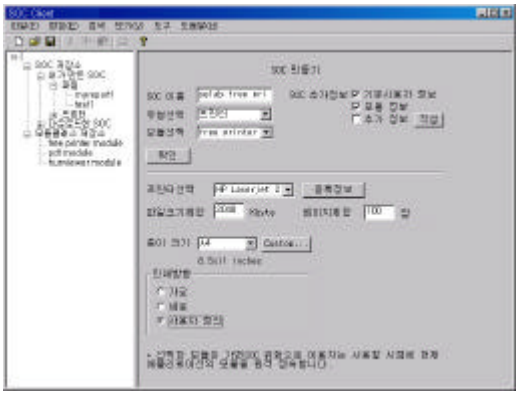

Figure 7. User interface for SOC creation

4.2 EVALUATION

Designed to measure the information retrieval time and network efficiency according to increases in clients, this experiment compared a pure P2P and the SOC based system proposed by the papers. Here are static element signs for the experiment. Variable element are port, processor, memory, IPC, resource type, and client number.

Static elements
M_n : Number of the management server
Bw : Network Bandwidth
M_{ps} : Maximum processor
M_m : Maximum memory
S_m : Module size

Here are static variables set up for the experiments such as M_n = 1, B_w = 10Mbps, M_{ps} = 50, M_m = 10Mb, S_m = 500Kb. Also M_m, and M_{ps} are set up to the application.

Following express T is an average resource sharing time.

$$T = \{(Cn \times (Cn-1)) \times (rc \times h)\} + \{(rc + st + t) \times (1-h)\}$$

yc : number of sucessed access by request

ys :number of allocate shared object when access to client

h : probability for service client, $yc/(yc+ys)$

rc : quantity of computing resource consumed by new shared resource

st :time of transfer resource consumed

Figure 8 show graph of performance according to increase in number of client. X axis of the graph is resource type each by sequentially increase number of client, Y axis is time cost by X axis. Figure 9 show graph of hit rate(success) by number of request between client and client each other

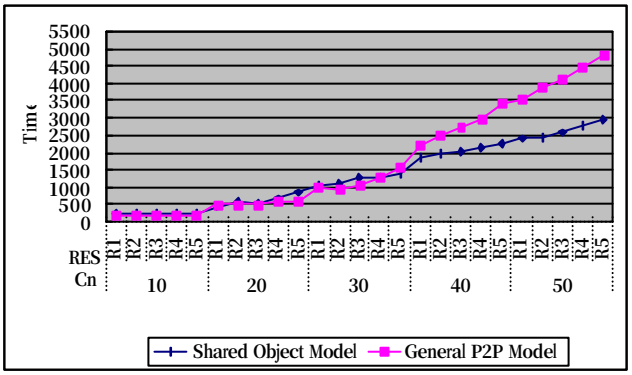

Figure 8. Graph of performance test according to increase in number of client

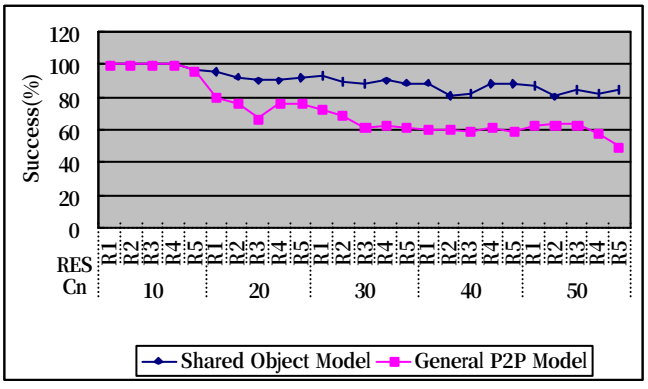

Figure 9. Graph of hit rate(success) by request from client

The outcome of the experiment shows that two models are

similar when there are not many clients. However it shows that the excellences of the shared object model as the number of clients and the number of types in the shared resources increase.

5. CONCLUSIONS

In this paper, we proposed the shared object management system, an object-based P2P architecture. The system which is designed based upon shared object model consists of platform layer, core layer, service layer, and application layer, so that each layer can be independently implemented and sustained. Also the system is designed and implemented using java interface, and supports the interoperability of the distributed objects through request relay manager supporting RMI over IIOP.

The proposed system can manage the resources afforded through a class by the name of SOC as objects, is so designed as to be extensible in diverse ways under P2P environments by means of distributable and pluggable functions.

Accordingly, it is applicable as a new model to computing power and efficient, distributive operation and management in building P2P in the enterprise information system. In addition, it shows a new possibility of P2P by expanding the scope of shared resources that includes hardware on a P2P network and services as associated with off-line, ASP, contents, and the like. The current dissertation also opened a new possibility that it is possible to operate P2P networks more efficiently by proposing a new algorithm for distributive network management.

In conclusion, it is verified that the shared object model more efficiently manages the given computing resources through using module classes and shared objects and through resource management of each application when the number of resource types increase. Accordingly by applying shared object model it is known that not only the efficiency and expandability in the resource sharing but also resource sharing capability of P2P applic ations increase.

As a further study and research project, a continuous study should be made on the ways of enhancing performance of the system service, ways of carrying out configuration of diverse agents for operation of more dynamic services, and on the expansion of pluggable functions offered as a basic requirement.

REFERENCES

[1] Endeavor Technology, "Introducing Peer-to-Peer," http://www.peer-to-peerwg.org, 2000.

[2] Andy Oram, "Peer-To-Peer," O'Reilly, Mar., 2001.

[3] G. Fox, "Peer-to-peer networks," Computing in Science & Engineering, Vol. 3 Issue: 3, May/Jun., 2001.

[4] John M, A. Roy, "Peer-to-Peer Internetworking Joins Browser-Based Computing," Merrill Lynch, 2000.

[5] U. G. Kang, H. J. Hwang, S. H. Lee and C, J, Wang, "SORBA:Shared Object Request Broker Architecture for Peer Computing," Proceedings of the IASTED International Conference Internet and Multimedia Systems and Applications, Aug., 2001.

[6] Munindar P. Singh, "Peering at Peer-to-Peer Computing," IEEE Internet Computing Vol.5, No.1 , Feb., 2001.

[7] Parameswaran, M., Susarla, A., "P2P networking: an information sharing alternative," Computer, Vol. 34 Issue: 7, IEEE , Jul., 2001.

[8] G. A. Bolcer, M. Gorlick, A. S. Hitomi, P. Kammer,B. Morrow, P. Oreizy, and R. N. Taylor. "Peer-to-Peer Architectures and the Magi™ Open-Source Infrastructure", Dec., 2000, http://www.peer-to-peerwg.org.

[9] Applied MetaComputing, "Legion – An Integrated Architecture for Secure Resource Sharing", 2000, http://www.peer-to-peerwg.org.

[10] HP, "e-speak Architectural Specification," Release A.0, http://www.e-speak.com, Dec., 2000.

[11] Project JXTA, "Technical Specification," Version 1.0, http://www.jxta.org, 2001.

[12] David Kuck , "Peer to peer and distributed computing ," ACM Press, 2000.

[13] Peer to Peer Working Group, "The Working Group on Peer-to-Peer Computing," http://www.p2pwg.org, 2001.