Middle-East Journal of Scientific Research 23 (9): 2302-2306, 2015
ISSN 1990-9233
© IDOSI Publications, 2015
DOI: 10.5829/idosi.mejsr.2015.23.09.22738

Distributed Model for M-Com Applications Using Delegate Object

[1]N. Shenbagavadivu, [2]I. Bremnavas, [3]M. Bhuvaneswari and [4]G. Geetha Ramani

[1]Department of Computer Applications, Anna University – BIT Campus, Trichy, India
[2]Department of Computer Engineering and Networks, Jazan University, Jazan, Saudi Arabia
[3]Department of ECE, Anna University-BIT Campus, Trichy, Tamilnadu, India
[4]Department of Mathematics, Anna University – BIT Campus, Trichy, India

Abstract: The aim of distributed delegate object model has an effect on performance of m-com (mobile commerce) business transaction by reducing the round trip time of huge number of transaction requests. Delegate object is a shared object which acts as a place holder for the valuable m-com business information. It also updates the customer information automatically, managed by the middleware applications. This model is tested thick client (mobile client) architecture with m-com business data.

Key words: Delegate object · Distributed · M-com business · Mobile computing

INTRODUCTION

The growing popularity of Internet technology and m-com (mobile commerce)business is increasing the demand for complete distributed application environment. In the present scenario of the m-com applications are widespread among medium and large-scale industries such as mobile money transfer. These applications are working highly distributed and heterogeneous environments. Certain issues that must be carefully considered during design and implementation of distributed applications. For example the client can access the business data directly may cause security problem. The updated information didn't retrieve in proper time because the data updating is performed in server side. A generalized distributed architecture model using delegate object is to improve the availability and effective data retrieval of information for different types of business clients. In the proposed model, shared object acts as a place holder for the valuable business data that can be shared by various distributed applications for reliable and synchronous data transmission which improves the security and content availability. Normally m-com business applications always have to provide an upto date information to their clients. This process requires constant querying of database by the application layer. Most of the m-com business data is held using a delegate object. The general distributed architecture uses an asynchronous pattern such as event base perform time consuming tasks.

Review of Literature: Ceglar *et al.* [1] discussed the concepts of a collaborative shared object framework and also described the concept can be incorporated in an existing object oriented toolkit. It's simplified through the elimination of redundant structures and the reduction of object instance. Lopes *et al.* [2] has developed a distributed environment which implements the notion of shared object, distributed shared objects and also introduces active shared mobile objects. It has simulated a distributed shared memory with better security, good performance, reliability and transparency. Green *et al.* [3] has described the design and implementation of the multi-tier portal architecture for thick client model. This multi-tier portal architecture is required in order to provide a scalable and flexible architecture for beginner to survive the complicated business environment. Janakiram *et al.* [4] initiated a new model named the surrogate object model to handle the asymmetry problem in distributed mobile systems. This model focussed an architecture that allows mobile devices to contribute seamlessly in computing and communication. It involves to bridging

Corresponding Author: N. Shenbagavadivu, Department of Computer Applications, Anna University – BIT Campus, Trichy, India.

between the mobile device and its support environment, using a place-holder namely the surrogate object. This is done by creating the surrogate in the static network to act on behalf of each mobile device. The surrogate object can remain active, maintaining information regarding the current state and plays an active role on behalf of the device. The surrogate object model elegantly solves a whole set of problems in distributed mobile systems: location management, mobile data access in client-server systems, disconnected operations etc. Kang *et al.* [5] has proposed that a framework to keep mobile application updated with changes in location maps which are stored in a distributed database server. This framework is suitable for application that requires frequent updates especially if new maps are added to the collection or database. The literature on distributed and mobile application environment indicates that frequent data retrieval from a remote database in a distributed mobile application requires a higher transaction time.

Structure of Delegate Object: The attribute new data always carries the latest information which is extracted from the business database. The attribute old data holds previous value from new data which is needed for comparison. Date attribute is used to notify the updation time. The attribute flag is controlled by data retrieval service and data updation service to hold the status of m-com business data.The structure of the delegate object is shown in Figure 1.

The getInstance() method is a static method which returns an instance of the delegate object. The set and get methods ensure the updation and retrieval of the attribute values. The update() method helps to check whether an information update has been completed or not and also update the flag value. The isupdate() method returns the current status of the flag attribute.

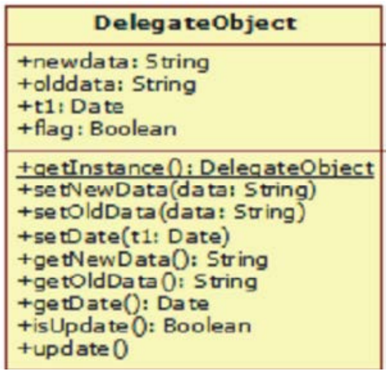


Fig. 1: Structure of the delegate object

Proposed Distributed Model: In this proposed distributed model the shared object acts as a place holder. Forreliable and synchronous data communication which improves the secured content availability in the valuable business data that can be shared by various distributed applications. The shared object is named as delegate object. The delegate object is responsible for notifying the client's nearby events, such as the insertion of new data (or) the modification of the existing data. When new data is updated, either by inserting new record or modifying an existing record,the database is informed about the event and gives the necessary information to the corresponding delegate object by the triggering of a function. The updated information is automatically transferred to the client through a delegate object. The delegate objects are managed by the middleware applications.

In the generalized distributed architecture various business services can be handled through delegate object. The data retrieval service forwards the client's request to a container. The container obtains the appropriate delegate object's reference by using a middleware application such as message oriented middleware, object request brokers and enterprise service bus.Delegate object acts as a place holder for the data. The data updation service transfers the data to the respective delegate object from the business database whenever the data is updated. The response content is clearly separated from the presentation layer since the client may be a web browser or a mobile device. The Figure 2 shows the generalized distributed architecture model using delegate object.

This model reduces the data access time since the client request is handled by the middleware application using delegate object. Otherwise the middleware application communicates to the corresponding database then the updated information is transferred to the client. It also ensures the integrity of the data between delegate object and database. The trigger service is used to achieve the data integrity. Performance of this model is evaluated against thin client and thick client architecture in distributed environment.

Thick Client Model Using Delegate Object: Most of the m-com business applications are required to be highly scalable. When business logic resides on the client side, it reduces the processing load on the application server or

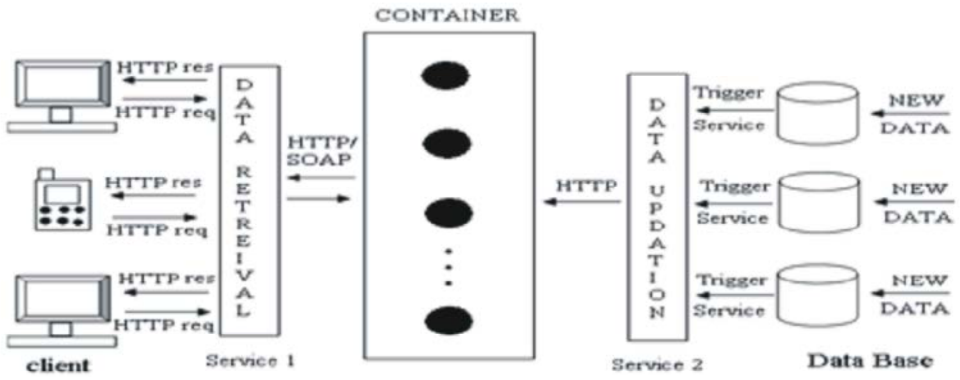


Fig. 2: Generalized distributed architecture model

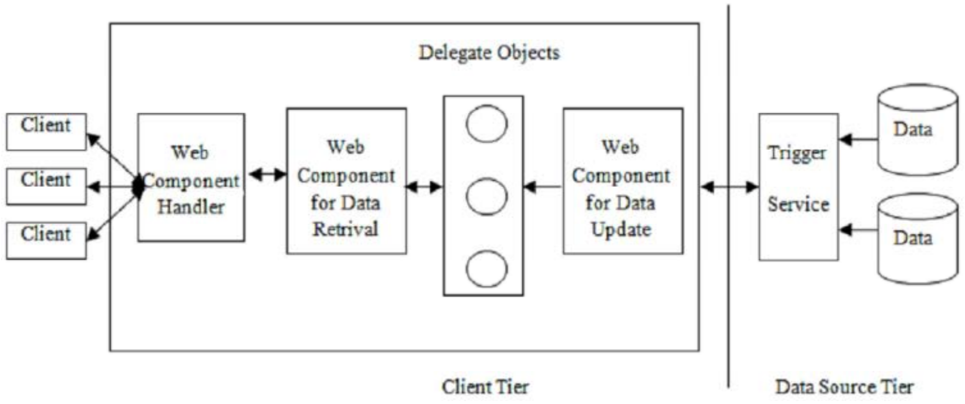


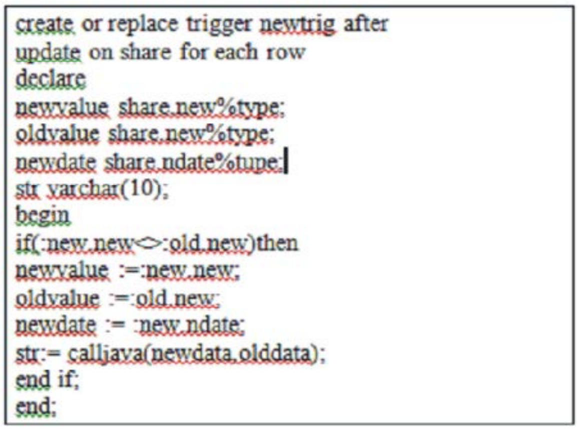
Fig. 3: The test bed of thick client model using delegate object

database thereby improving scalability and better load balancing. Most of the e-business applications are today m-com business applications (mobile). Mobile applications run on limited resources. Therefore business logic and presentation logic are implemented as a web component. This web component also acts as a manager to manage and communicate with the delegate object. The model is shown in Fig. 3.

In this model, the delegate object is deployed in the web container. When the client sends a request to the handler, it takes the responsibility of providing the requested service. The updated service information is held in the delegate object. The web component for data retrieval retrieves the data from the delegate object and sends back as a response to the handler. The end client may be a browser or mobile device, so the result is generated according to the client type. The handler customizes and processes the data received from the web component according to the type of the client. Any changes or updation made in the database is monitored and triggered service is invoked. The delegate object is then updated by the data updation service.

Communication Model for Thick Client Architecture: In this model, the client uses web browser or mobile application to send a request for accessing a particular business service. To access a service from the server the client can register himself with the server just by invoking the registration procedure. The server obtains the necessary information from the registered client and then it sends a list of available e-business services as a response. The client selects a service from the available list and invokes the same. The client request is then passed through a custom pipeline which contains a collection of web components.

To process the request, the data retrieval component finds a reference for an appropriate delegate object that is associated with the requested service. The service provider, which is called data update web component, transfers the newly updated data to the respective delegate object from the business database through a trigger service. The trigger service strongly noticed whenever there is a change of business data in the database change. The important features of this model are that, it reduces the data access time and synchronizes the



```
create or replace trigger newtrig after
update on share for each row
declare
newvalue share.new%type;
oldvalue share.new%type;
newdate share.ndate%tupe;
str varchar(10);
begin
if(:new.new<>:old.new)then
newvalue :=:new.new;
oldvalue :=:old.new;
newdate := :new.ndate;
str:= calljava(newdata.olddata);
end if;
end;
```

Fig. 4: Trigger service

data transfer between database and the client through web components. The delegate object can be modified and accessed any number of times by the web components. The data retrieval web component obtains the data from the delegate object and transfers the response to the handler.

Implementation Details of the Delegate Objects in a Thick Client Model: The generalized thick client model is implemented in Java. The web components are Java servlet programs. The m-business data are maintained in the Oracle database. The data updation is carried out using SQL statements. The following database trigger shown in Figure 4 is fired when a modification is performed in the database.

The calljava() function invokes another static function send() which is written in Java. As the function is static, it can be involved without creating a reference to an instance of the class object in which it is defined. The class file is loaded into Oracle using Load Java utility. On loading the file, two new tables are created automatically in the database. The table JAVA$CLASS$MD5$TABLE is used to store the reference of the class files that are added and the table CREATE$JAVA$LOB$TABLE is used to store the class files itself [Sujatha2009].

The new data, old data and the updated time are handled by the static function. This function establishes a link to the service provider (web component for data update) using URL and URLConnection classes. The service provider gets the instance of the corresponding delegate object using getInstance() method and updates the new and old data using set methods in DelegateObject class. It also modifies the flag attribute value as 'true' using update() method.

Performance Analysis: Round Trip Time (RTT) is used as a performance metric to evaluate the implemented distributed m-business system. The major factor that influences the performance of the delegate object model is RTT. The RTT measures the time needed from the point when the client initiates a request to the point to the client receive the response.

Experimental Configuration in Thick Client Model: Two testing applications are developed to analyse the performance of the e-business system. These two tests are:

Test A: Performance analysis of thick client model without delegate object.
Test B: Performance analysis of thick client model with delegate object.

In Test A, client requests are processed by web component. The web componentin turn communicates with the database and retrieves the updated value and returns to the client. Since a new data or updated data is to be retrieved each time, the web component has to send a new request to the database. The web component or the client is unaware of the changes happened in the database. Therefore, this model uses a polling pattern to place a request to the database at frequent time intervals. The round trip time is measured for all the client requests.In Test B, the performance is evaluated for thick client architecture using a delegate object. The delegate object and handling functions are deployed in web server. Whenever the client request is initiated the corresponding delegate object is accepted and it processes the request by forwarding the response to the client.

The RTT is measured for all the client requests. In both the models, the initial RTT time is higher as the web component establishes a connection with the database. The further queries are implemented asynchronously thereby reducing the RTT value. In the above two cases, the RTT is measured in terms of milli seconds for a specific m-business system and plotted as a graph which is shown in Figure 5. The average RTT is evaluated in terms of milli seconds which is shown in Table 1.

The average RTT is evaluated in terms of milli seconds which is shown in Table 1.

By comparing the RTT computed for both the models it is determined that using delegate object model's average RTT time is less than the model without using delegate object.

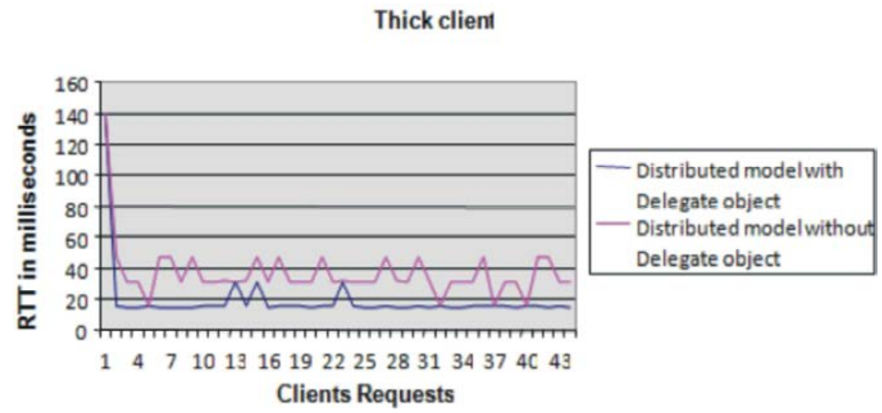


Fig. 5: Performance of RTT – thick client model

Table 1: Average Round Trip Time in milli seconds

Distributed Model	Average RTT in milli seconds
Distributed model with delegate object	19.43182
Distributed model without delegate object	36.59091

RESULTS AND DISCUSSION

The major factor that analyse the performance of the delegate object model is RTT. The average RTT of thick client and thin client model is analyzed. It is determined that using delegate object model's average RTT time is less compared to the model without using delegate object. The proposed delegate object model is considered as one of the best possible alternatives for m- business environment since it follows asynchronous communication patterns.

Conclusion and Future Work: In this paper, we introducedthe delegate object model is developed for distributed environment to enhance the information retrieval in m-com business applications. The delegate object model has an effect on the performance of m-com business transaction by reducing the network overhead of huge number of transaction requests. This is tested on thick client architecture and their performance evaluation is carried out with respect to RTT between the request and response. The developed distributed models can be tested with large scale m-com business environment in real time for portability across different service providers and heterogeneous network scenario. In future, the proposed work can be extended in heterogeneous distributed mobile environment.

REFERENCES

1. [Ceglar 2001] Ceglar, A. and P. Calder, 2001. A New Approach to Collaborative Frameworks using Shared Objects. Proceedings of the 24th Australasian Computer Science Conference, 2001, ACM International Conference, pp: 3–10.
2. [Lopes 1996] Lopes, D., S. Hammoudi and Z. Abdelouahab, 1996. Parallel/Distributed Programming on the Web. IEEE International Workshop on Database and Expert System Applications, pp: 291-312.
3. [Green 2007] Green, M.L. and S.D. Miller, 2007. Multitier Portal, Architecture for Thin- and Thick-client Neutron Scattering Experiment Support, International Workshop on Grid Computing Environments.
4. [Janakiram 2005] Janakiram, D., M.A.M. Mohamed, A. Vijay Srinivas and M. Chakraborty, 2005. SurrogateObject Model: Paradigm for Distributed Mobile System', in ACM International Conference on Information Systems Technology and its Applications, pp: 124-138.
5. [Kang 2007] Kang, H.K., K.J. Li and M.S. Kim, 2007. A Framework for Dynamic Updates of Map Data in Mobile Devices., International Journal of Web Engineering and Technology, 3(2): 176-195.
6. [Sujatha 2009] Sujatha, S., A. Krishnan and V. Ramachandran, 2009. An Enhanced and Secure Messaging Architecture for a Distributed e-business Environment, International Journal Electronic Business, 7(3): 287-300.