Cooperation Model for Object Group using Load Balancing

Romeo Mark A. Mateo, Insook Yoon and Jaewan Lee

School of Electronic and Information Engineering, Kunsan National University
68 Miryong-dong, Kunsan, Chonbuk 573-701, South Korea

Summary
Currently, object implementations facilitate object group model to organize and manage efficiently the object services. In addition to the complexity of implementing the object groups, researchers also considers the quality of service (QoS) by various means such as replicating the services within the system. Load balancing scheme is necessary for efficient load distributions within the replicas. Two methods commonly used which are the round-robin and least load algorithms where each concerns are fast response time and efficient load distribution, respectively. On the other hand, researchers focus on adaptive load distribution where it considers both. However, object groups require the interaction of each component to cooperate in implementing the efficient load balancing. This research proposes a cooperation model for object groups to implement the load balancing. Each component from the object group is designed to coordinate in the load balancing. The adaptive scheme is based on two algorithms which are the round robin and the proposed fuzzy least load. Fuzzy least load algorithm is used to determine the least loaded object processed in the fuzzy system. For optimizing the proposed algorithm, the fuzzy sets are trained from the previous data. The efficiency of load distribution is measured by means of calculating the total client request completion time.

Key words:
Distributed object groups, load balancing, fuzzy logic, neuro-fuzzy system.

Introduction

Common Object Request Broker Architecture (CORBA) is mostly used for robust implementation of objects in distributed environment [1]. CORBA has been standardized for several years by the Object Management Group (OMG) which is a non-profitable organization and concerns with the improvements of object technology based on trends in the industries. There are lots of researchers in CORBA expanding the implementations to web services, ubiquitous paradigm, mobile services and intelligent services. In large enterprises, most of the issues focused on acquiring the system's QoS and managing the critical transactions. The QoS is acquired by variety of techniques and one method is implementing replication scheme. Replication is a key to achieve high availability and fault tolerance in distributed systems [2]. It is often achieved in CORBA-based distributed systems by using replication of services [3]. Replication is a technique for enhancing services by assigning the workloads to each replicated objects which makes the services more available. Fault tolerance is also achieved in replication by making a copy of the object so that client may be able to access data at an alternative server if the default server fails or become unreachable. According to these necessities, managing the replicated objects is necessary by means of object group models. Object group models are designed to manage the system by grouping the appropriate objects needed to operate the whole service. Also, object group modeling is one concentration of research in recent studies on object services management and various approaches are used to implement effective object groups [4]. The effective management to the object group is critical for coordination of the objects.

In replication scheme, it is important to have an efficient distribution of the system loads while the system is at run-time to achieve the QoS. Also, efficient algorithm is important to implement the load balancing scheme in object group models. Recent research studies are focusing on genetic and evolutionary algorithms but these are more complex to implement in CORBA [5]. Round robin and minimal dispersion methods are commonly used for load balancing the objects. Othman introduced the adaptive load balancing in CORBA [6, 7]. An intelligent load balancing in Jini implementation is presented [8]. It uses the fuzzy concept on deciding the appropriate service. However, the schemes are efficient only in servers having a single replicated object and does not deal with load balancing the object groups.

The contributions of this research are the proposed cooperation model for the object groups and the implementation of the load balancing. The proposed healthcare expert system is an online health consultation system which it takes the advantage of the object group technology. Moreover, the object groups cooperate to implement the adaptive load balancing scheme which is based on two algorithms; round robin and the proposed fuzzy least load algorithm. The system analyzes the equal distribution of loads and it switches the algorithm to adapt

This research was supported by the Program for the Training of Graduate Students in Regional Innovation which was conducted by the Ministry of Commerce Industry and Energy of the Korean Government

the system requirements based on the threshold value. Fuzzy least load algorithm is used to determine the least loaded object based on the fuzzy logic rules. To have an accurate classification, the fuzzy sets are trained from the previous data by the neuro-fuzzy algorithm. The efficiency of load distribution from the group cooperation model is measured by means of calculating the total client request completion time.

2. Related Works

2.1 CORBA Object Group Modeling

The Object Management Group (OMG) which was created in 1989 solicited the input from all segments of the industry and eventually defined CORBA standards [1, 9]. CORBA specification has been implemented by numerous hardware and software system manufacturers, creating a rich and robust framework that successfully operates across heterogeneous computing platforms. The specifications focus on object technology with a client-server model to provide a uniform view of an enterprise's computing system and everything in the network is an object. The necessity to manage these objects is modeled in the object groups to perform cooperatively for an efficient service is realized by many object designers and programmers. There are researches integrating their methods like the real-time CORBA [10] in the object group model for real-time services. The object group modeling was suggested for providing the efficient and convenient management of distributed objects, and for making the individual object independent and reusable. To define formally, an object group is a set of objects related logically. A group acts as a logical addressable entity where an entity that requests a service from a group is a client of the group. The properties of the object groups [4] are shown below.

1. Group behavior – A set of action taken by the members of the group. This encompasses the way the member of objects are coupled and how they cooperate.
2. Group reference – Acts same as an object reference where it designates a single object while group reference designates a set of objects. The number and location of these objects is hidden from the entity that holds the reference and may vary over time.
3. Group membership – This determines the group where the object belongs. This property is important to organize the procedure of a service and the management of the objects.
4. Encapsulation of groups – This is a property of an object group to behave like a singleton object where it can act as an identifiable, encapsulated entity that may be invoked by a client.

2.2 Load Balancing Schemes

Implementing load balancing and fault tolerance based on replication schemes to the distributed objects promotes QoS, scalability and dependability through the system. Several middleware-based load balancing schemes are studied by researches like the research of Othman [6, 7] where the performance of round-robin and minimal dispersion (least load) load balancing schemes are analyzed. Using the round-robin provides a fast forwarding of request and has fast response time to clients request but the distribution of loads from the replica may not be equal. The disadvantage of the round-robin is solved by the minimal dispersion algorithm but it consumes time on determining the least loaded server where there is latency on the response time. He also carefully discussed the challenges of implementing the load balancing service and illustrated the components which are necessary. Finally, he proposes the adaptive scheme where the two algorithms are used to implement the efficient load balancing. However, the scheme only considers a single object in a replica server while the trend of object technology is in the group of objects where there is a need to consider the different properties ofobjects for implementing the load balancing. Load balancing the distributed object group using a dynamic replication [11] is proposed for efficient and fast service. The load balancing service uses the dynamic replication scheme which is mechanized by flow balance assumption that derives from the arrival and service rate to execute request forwarding to new objects. The scheme shows the improvement of the mean client request completion time of the system as compared to other load balancing schemes. However, the scheme is only limited on using the round-robin where the object group model needs the efficient load distribution by means of monitoring the loads and determines the least load to forward the next client request. This method is important to balance the distribution of the clients request and also improves the performance of the system. Fuzzy logic controller is used in load balancing the Jini services [8]. The approach works by using a fuzzy logic controller which informs a client object to use the most appropriate service such that load balancing among servers is achieved. Jini is used to simulate the middleware platform, on which the proposed approach and as well as other approaches are implemented and compared. Like in the research of Othman, the algorithm also only considers a single object in a server.

2.3 Fuzzy Systems

Fuzzy classification is based on the concept of fuzzy logic and fuzzy sets, which was conceived by Lotfi Zadeh [12]. It is presented as a way of processing data by allowing

partial set membership rather than crisp set membership or non-membership. Fuzzy logic is one of the earliest concept using fuzzy sets where most fuzzy controllers benefit this technique because of the it's smooth transition to transfer the another fuzzy set and the way it handles its imprecise values. Figure 1 shows an example of a fuzzy logic controller where used in load balancing the Jini's distributed services [8]. Fuzzifier is an interface for the input where maps the numeric input to a fuzzy set and it maps the premises of the fuzzy rules from the rule base. The fuzzy inference engine applies the inference mechanism to the set of rules using the membership function. Lastly, the crisp value is determined by the deffuzifier.

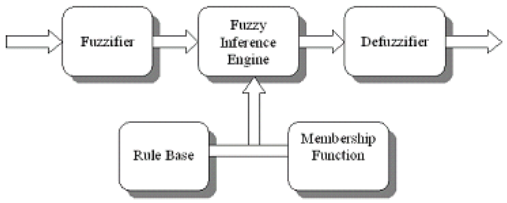


Fig. 1. The components of fuzzy logic controller

Fuzzy concept is also a popular tool for information retrieval [13] where it is derived from classical methods. Fuzzy c-means classifier (FCM) uses an iterative procedure that starts with an initial random allocation of the objects to be classified to c clusters. Among other data mining techniques, FCM is a very popular tool for knowledge extraction in the distributed environment. Fuzzy algorithm like FCM is used for filtering the neural networks [14] and improved the speed of data processing. The filtering method is processed by grouping the data, selects the most relevant of it and removes the unnecessary data and consequently and more likely it improves the quality of rules or information extracted.

The improvements of fuzzy systems become more adaptive to the changes of the system requirements by adjusting the fuzzy sets for classification and one of these are the neuro-fuzzy systems. There are many types of neuro-fuzzy system used for automatic controls and most of them used the Sugeno-Type [15]. The S-T neuro-fuzzy system is used for a scheme to construct an n-link robot manipulator to achieve high-precision position tracking [16]. Also, after the implementation in control systems, the neuro-fuzzy were used for data analysis and fuzzy rule mining [17]. Typical fuzzy data analysis discovers rules in large set of data and these rules can be used to describe the dependencies within the data and to classify a new data [18]. A neuro-fuzzy classification (NEFCLASS) is a fuzzy classifier that creates fuzzy rule from data by a single run through the data set [19, 20]. The procedures are defined below.

1. Learning the fuzzy rules – domain attributes are mapped to the units of the input layer and output layer of the neural network which contains one unit for each possible value of the class attribute. This procedure creates fuzzy rules and adapts the fuzzy set appearing in the rules of learning.
2. Learning the fuzzy sets – fuzzy sets are learned by training. The fuzzy sets adjust based on the constrained parameters of the system. After the training of the fuzzy sets, the best rules are determined based on the fuzzy sets and normally fuzzy rules from the hidden node are reduced depends on the parameters defined.

3. Cooperation Model for Object Group

Cooperation is the behavior of the system components working together for attaining the same goal. We propose the cooperation model for the object groups to implement the load balancing. The architecture of the proposed healthcare expert system (HES) shown in Figure 2 is designed in object group model.

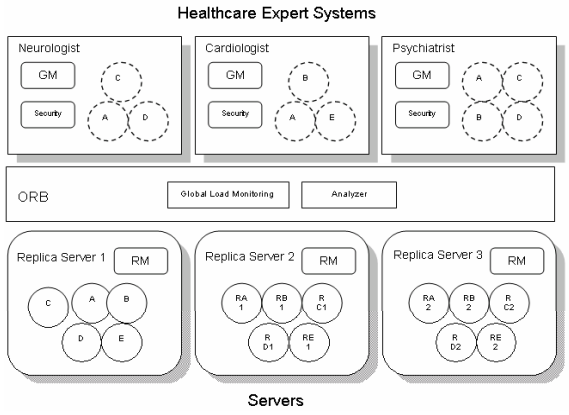


Fig. 2. The object group model for the healthcare expert system

On the top level, consisting of services where having the object references of each object it needed to function. This service have sub-component which resides in the server replica shown in the bottom level of the architecture. Services, by acquiring the property of choosing and managing the objects, are considered as an object group in this research. Middle level part consists of global load monitor and analyzer which manage the load distributions. On the bottom level of the system consists of replica servers. A server replica is managed by the replica manager (RM). A server replica refers to a single server contains variety of objects. Communication is managed by the group managers or GMs through the object request

broker (ORB). Figure 3 shows the interaction of client to the service. Group managers (GM) receive the request and process the information through the sub-component objects. After processing, a return output of consultation is send to the client.

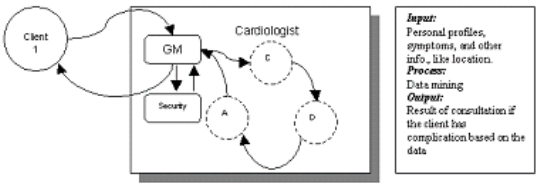


Fig. 3. Client consultation to a single service of HES

3.1 Group Components

1. Group manager (GM) – the main component of the service. This does all the management procedure. All the components are connected to the GM and other GM from outside the group are coordinating.
2. Security – checks the validity of the client. The security verifies if the client has the access privilege to the object. The proposed system processes this with the encryption module of the security. If the client is not allowed to access the object then it sends a message that it requires a valid user id and password or the request is not allowed at all. After the procedure, it sends message to GM the result of verification.
3. Service – composite of the task which consists of a several object services to provide the service to client. In this research, a service is also considered as an object group. The service consists of GM, security and object services.
4. Replica manager (RM) – does the creation and management of objects in the server. The RMs within the server replicas is coordinating to other RM to manage the replicated objects. If object that changed it values then RM of that object communicates through other RM to change the value of the same object
5. Object services – sub-components of the services. These sub-components are requested by the GM to process the request. Moreover, the object references are managed by the GM to be grouped. The purpose of grouping the objects is to gather the related object based on the properties of the service. In the group model of objects, each group uses a plurality of request. This is a property of an object group to behave like a singleton object where it can act as an identifiable, encapsulated entity that may be invoked by a client.
6. Global load monitor – monitors the loads. Each replica is registered to the global load monitor. Upon receiving the loads, the global load monitor checks the value of the current distribution to compare on the threshold value. If the load distribution is greater than the threshold, then it communicates to the GMs to switch the scheme from round robin to fuzzy least load algorithm. The same procedure of switching from fuzzy least load to round-robin is done if the global load monitor determines that the load distribution is lower than the threshold.
7. Analyzer – analyzes the load of each servers and objects. This is a sub-component of the load balancing service which intercepts every request to process the fuzzy least load algorithm for distribution of request to the replica objects and determine additional object replicas needed. By processing the values to the fuzzy system, it determines the appropriate least loaded object to forward the request. Finally, it sends the object reference and issue a location forward.

3.2 Cooperation of the Object Group Components

The load balancing service consists of interactive components for intercepting and forwarding request. GM is the main receiver for client's request. The default scheme of the load distribution is round robin. Each object groups or services are aware of the current scheme it uses by sending message of their status. Once the service receives the request, it sends messages to other services. The content of the message is location of the objects and server replica which it invokes currently. This method makes the other services to be aware of the objects currently invoking. If another service receives a request then it begins invoking the object that is not used by other object groups. Figure 4 shows the communication between the services.

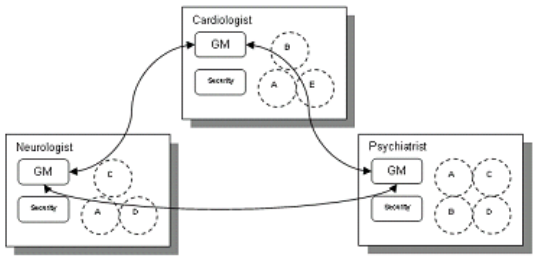


Fig. 4. Cooperation of the object groups

Upon the request, the object sends an update loads to the global load monitor. An increment of loads is done to the global load monitor and after the servicing the client, it sends again an update load which decreases the load. While the global load monitor is on the update procedure, it compares the status of the load distribution from the objects if it already exceeds the threshold value. More

discussion of getting the threshold and load distribution are explained in Chapter 4. Figure 5 shows the updating of loads from the objects of the server replicas to the global load monitor.

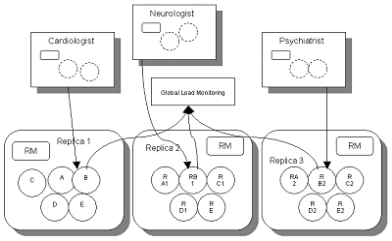


Fig. 5. Updating the loads to the global monitor

After invoking the object by the GM, the object sends a load update to global load monitor. If the value of load distribution is greater than the threshold then the round robin scheme is switched to the fuzzy least load. At this point, request of GM is directed to the analyzer where it determines the suitable object to and procedure is done in the fuzzy system. Analyzer issues LOCATION_ FORWARD() reply to the GM request with the reference object. Client sends the request to the new reference. Figure 6 shows the interaction of the load balancing components.

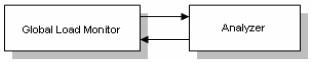


Fig. 6. Interaction of the load balancing components

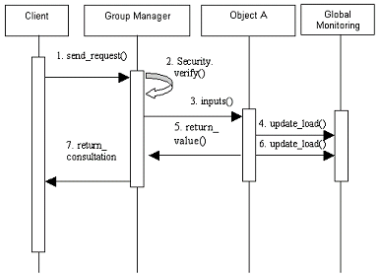


Fig. 7. Event trace diagram (ETD) for round robin where $\sigma < \Phi 1$

Figure 7 presents the procedure of the round robin algorithm. Round robin is the default scheme of the system while. The σ determines how well the loads are distributed to the objects and Φ_1 is the threshold value for the load distribution. Chapter 4 explain the details of getting the σ and Φ_1. When the load distribution is less than the threshold on the round robin scheme, it implies that there is no need to forward the request because the system resources are utilized enough in distributing the loads. The switching of algorithm occurs only if the load distribution is greater than the threshold which means that the variance of the load distribution is large. Figure 8 shows the case of where the system switches to fuzzy least load algorithm.

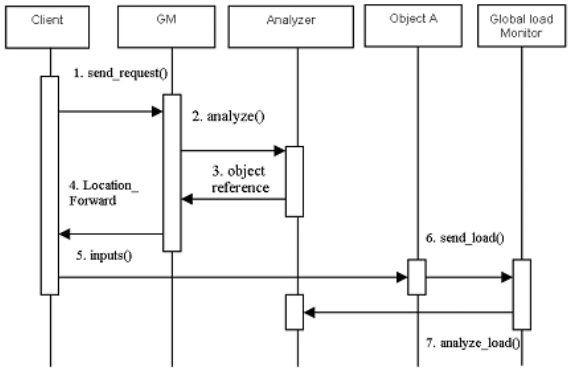


Fig. 8. Event trace diagram (ETD) for round robin where $\sigma > \Phi 1$

Figure 8 presents the ETD of analyzing the loads from each server and the loads of objects. Server loads refers to the number of current requests accessing the objects in the server while the object loads refers to the current request accessing the object. These loads are analyzed in the fuzzy system which consists of Fuzzifier, Rule Base Membership function, Fuzzy Inference Engine and Deffuzzifier. The fuzzy rule base contains a set of linguistic rules with linguistic values and variables. Different linguistic values can be assigned to linguistic variables. For example MOST, MEDIUM and LEAST is used for variable of candidate object. Based on the linguistic values, their corresponding membership functions are expressed based on application requirements. After analyzing the most candidate object, the analyzer sends location forward to the GM with the object reference of the most candidate object. In fuzzy least load scheme, the same procedure of updating the global load monitor is done. The analyzer continues to compare the threshold to the load distribution and when the threshold is greater the load distribution then the system scheme switches again to round robin.

4. Adaptive Load Distribution

The object groups cooperate to function the adaptive switching of load distribution for implementing the efficient balancing of load distribution. The system is adaptive to change the scheme from round robin to fuzzy

least load algorithm. Moreover, the fuzzy system is usedd to process the least load and determine the most suitable replica for the request. In addition, the classification of fuzzy least load is optimized by using the fuzzy set training of neuro-fuzzy algorithm based on the previous data. Figure 9 presents the procedure of the proposed fuzzy least load algorithm.

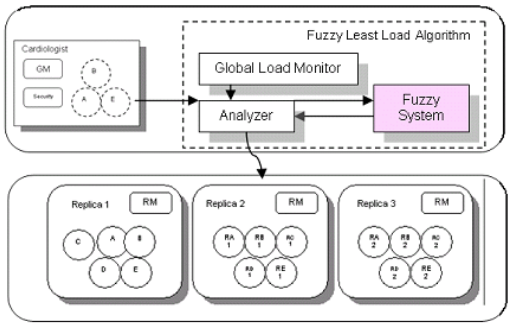


Fig. 9. Procedures of the load distribution in fuzzy least load algorithm

The algorithm is operational if threshold value (Φ_1) is less than the load distribution (σ) and the GM direct the request to the analyzer. In this research, the load is defined as the number of request currently processed by the object. First, the global load monitor determines the loads of each object and server replica where the load value is consist of the number of current load of the server. In other studies, [6, 7, 8, 11] they use one object per server. This research is more concerned on object group load balancing and assumes that the performance of an individual object is different to others.

4.1 Load Distribution

In the first phase of the scheme, round-robin takes effect. The round-robin is a scheme which directs the GM to the objects alternately within the servers. GM already has the knowledge which objects it starts accessing by the time it begins servicing the clients. The switching of the system from round-robin to fuzzy least load is determined by getting the distribution of the loads from the objects presented by σ and compared to the threshold value Φ_1. Equation 1 shows the threshold of the load distribution where α is set by the system administrator indicates how the system tolerates the distribution of loads to the replicated objects.

$$\Phi_1 = \alpha \tag{1}$$

Equation 2 calculates the load distribution which derives from standard deviation. It uses the variables from the number of the current loads in an object n (R_n) and its mean service time (μs_n). The value of σ increases when the distribution of the loads from different objects is not equal in performing round robin algorithm. A zero value from σ means there is equal distribution from the objects.

$$\sigma = \frac{\sum_{i=1}^{n} ((R_n * \mu s_n) - \mu)^2}{N} \tag{2}$$

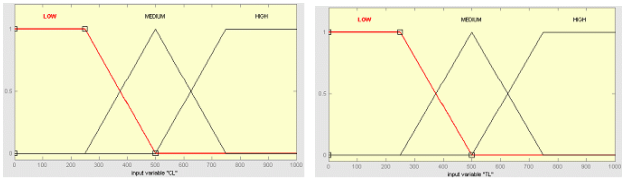
Equation 3 is the condition of Equation 1 and 2 to switch the round robin to fuzzy least load algorithm and Equation 4 shows the condition of switching fuzzy least load algorithm to round robin again. While the system operates the fuzzy least load scheme, the results are gathered and stored in the database with the response time from the GM. This is used to train the fuzzy system based on the performance of each request.

$$if\ \ \sigma < \Phi_1\ then\ \ \ switch\_fleastload() \tag{3}$$

$$if\ \ \sigma > \Phi_1\ then\ \ \ switch\_roundrobin() \tag{4}$$

4.2 Fuzzy Least Load Algorithm

We propose the fuzzy least load algorithm which is based on minimal dispersion [6, 7] or least load algorithm and implemented using fuzzy system. This research defines the load as the number of request currently in the object. The current load is the number of request in the queue of the object (OL_i) and the server load is the total number of load in the server (SL_i). These values are sent to the analyzer and processed to the fuzzy system. These two parameters use the fuzzy rules in Table 1. The fuzzy system determines the candidate object indicated by C_i where the procedure returns a crisp values on processing the OL_i and SL_i, and then the candidate which has the highest value on processing the center of gravity [12] is selected as the least loaded object. Neuro-fuzzy is used on the training procedure to adapt the fuzzy sets. The fuzzy sets of the two inputs in Figure 10 are linguistic variables. The object load and the replica server load in Figure 10 both have three linguistic values which are LOW, MEDIUM and HIGH load while the candidate is shown in Figure 11 have five linguistic values which are MOST, MEDIUM MOST, MEDIUM, LEAST MEDIUM and LEAST.



(a) (b)

Fig. 10. Fuzzy sets of the OLi (a) and SLi (b)

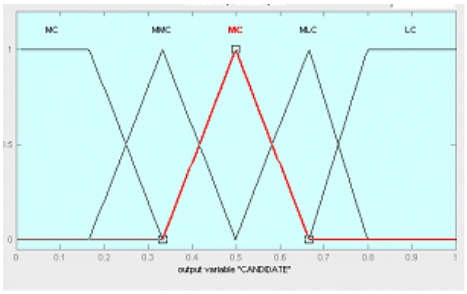


Fig. 11. Fuzzy set of Candidate

The candidacy value in Figure 11 is presented in an increasing manner because of the least constraint from the parameters must have a high candidacy of the object replica for being the least load. The complement from membership value is used shown in Equation 5. Equation 6 shows the conjunction of two variables process in minimum function and Equation 7 calculates the deffuzzified value of a rule.

$$\mu_{\neg A}(x) = 1 - \mu_A(x) \tag{5}$$

$$\mu_{A \wedge B}(x) = \min\{\mu_{\neg A}(x), \mu_{\neg B}(x)\} \tag{6}$$

$$R_i = conjuction\{\mu_{A \wedge B}(x_i), \mu_{A \wedge B}(z_i)\} \tag{7}$$

The proposed fuzzy least load algorithm uses the fuzzy rules from Table 1 to process in the fuzzy inference engine. These fuzzy rules in Table 1 are consisted of nine combinations from lower to higher value. The output value of a rule is calculated by the getting the complement function in Equation 6 of two fuzzy sets. Moreover the crisp value is determined to have a specific real value on determining the candidate for the least load. The real value is determined by using the defuzzification method of the center of gravity method in Equation 8.

Table 1. Fuzzy rules used in the proposed fuzzy least load algorithm

Rule	Object load	Server load	Candidate
1	LOW	LOW	MOST
2	LOW	MEDIUM	MEDIUM MOST
3	MEDIUM	LOW	MEDIUM MOST
4	LOW	HIGH	MEDIUM
5	HIGH	LOW	MEDIUM
6	MEDIUM	MEDIUM	MEDIUM
7	MEDIUM	HIGH	MEDIUM LEAST
8	HIGH	MEDIUM	MEDIUM LEAST
9	HIGH	HIGH	LEAST

$$C_i = \frac{\sum_{j=1}^{r} \mu_j \cdot s_j}{\sum_{j}^{r} \mu_j} \tag{8}$$

C_i is the final value where r indicates the number of rules. The replica object that has greater C_i value indicates that it is the candidate for the least load. After determining the least load replicated object, analyzer sends the object reference of the replicated object and issues a LOCATION_FORWARD() to GM. The pseudo code is shown below.

```
Ci = Replica load value
Ri = value of rule by calculating the conjunction function
for each μ(i) do
 for each input xj do
   begin
    'μ(i)ji = 1 - μ(i) (xj);
   end;
for each Ri do
   for each 'μ(i)ji do
   begin
     Ri = conj(Ri , 'μ(i)ji);
   end;
     Ci = ΣRi * si / ΣRi ;
     for each Ci do
   begin
    if Ci is the most candidate
      then choose the Ci to forward the request;
   end;
```

Fig. 12. Pseudo-code of the fuzzy least load algorithm

4.3 Learning the Fuzzy Sets

The neuro-fuzzy algorithm is used to train the fuzzy sets of the proposed fuzzy least load algorithm. The neuro-fuzzy system is trained based on the data from the history of successful classified forwarding of replicated object and its response time shown in Figure 13. After gathering the data, the fuzzy system trained offline to adjust the fuzzy sets of the each linguistic variable of the object and server so the load distribution method of the proposed algorithm is optimized.

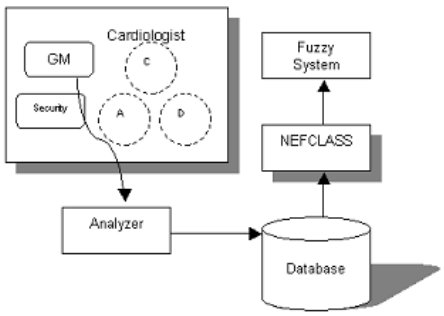


Fig. 13. Neuro-fuzzy classification algorithm

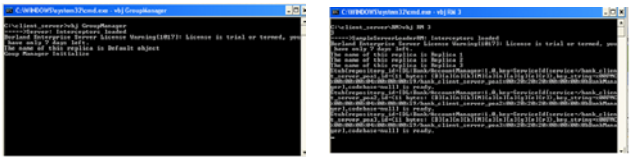
Before processing the current fuzzy sets to the neuro-fuzzy algorithm, the data are pre-processed first. The performance evaluation of the preprocessing is based on the response time to return the request back. A threshold is specified to choose only the pattern that is needed for the training the fuzzy set. The training pattern is arranged by the fuzzy variables by the load of the server and load of an object. After arranging this pattern, it feeds in the structured network of the neuro-fuzzy classification (NEFCLASS). The supervised learning algorithm of NEFCLASS to adapt its fuzzy sets runs cyclically through the learning set until a given end criterion is met. After a pattern is propagated, the error is determined for each output unit. Based on this error, for each active rule unit it is decided whether its degree of fulfillment should be larger or smaller. The membership function that is responsible for the degree of fulfillment is identified and only this fuzzy set is adapted accordingly. The learning of rules creates an additional node for U_2 and additional weight connection from U_1 and U_2 to map the inputs. This is done by selecting the next pattern (p, t) from learning task or L where L = $\{(p_1, t_2),\ldots, (p_s, t_s)\}$. For each input unit $x_i \in U_1$, the membership function is mapped. If there are still less than *k*max rule nodes, and there is no rule node R with $W(x_i, R)$ then create such a node c_l if $t_l = 1$. The fuzzy sets from the linked connection are also learned from data. The delta value is determined by Equation 9.

$$\delta_R = o_R(1-o_R)\sum_{c\in U_3} W(R,c)\delta c \qquad (9)$$

The adjusted linked connection from learning method using the delta value to apply the changes in the shared weights is used to classify the input data. The fuzzy sets are learned based on the iteration method of the NEFCLASS. After the training, the adjusted fuzzy set is used for the fuzzy least load algorithm.

5. Experiment Evaluation

The experiment used the Borland Visibroker 7.0 to implement the proposed system which is CORBA compliance. The object group and the load balancer component were developed in Java. The portable object adapter and portable interceptor were used to implement the load balancing service. The proposed fuzzy least load is encoded in the analyzer and group manager access the interface of analyzer every time the load distribution is greater than the threshold. Figure 14 presents the screenshots of the executing the group manager (a) and replica manager (b). Figure 15 shows the result of monitoring the object loads by the global load monitor.



(a) (b)

Fig. 14. Execution of the group manager (a) and replica manager (b)

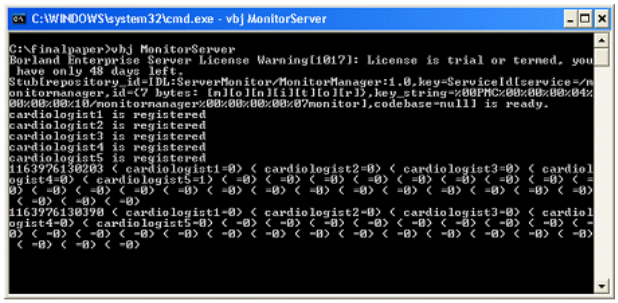


Fig. 15. Monitoring the object loads by the global load monitor

The proposed fuzzy least load (FLL) algorithm was compared to other algorithm namely, the round robin, least load [6] and on-demand replication scheme [11] through simulation. Measuring the performance is discussed in Section 5.1. Moreover, the fuzzy sets of the proposed algorithm are trained by using the neuro-fuzzy to have the optimal value of distributing the loads.

5.1 Performance Measure

The schemes are evaluated by adding all client request completion (C(s)) and divided by the number of clients (C). Equation 10 shows the calculation of C(s) where the transfer time of message request and service time are added.

$$C(S) = ST + TT(TServer + TClient) \qquad (10)$$

The transfer time is consisted of transfer latency of the request message from the client to the server and transfer latency of the result message from the server to client. The mean client request completion ($\mu\_C(s)$) is given in Equation 11. The capacity of the system to process the request is measured by Equation 11. Processing all clients request in a lesser time means the scheme is more efficient. The efficiency of the scheme is determined by increasing the number of clients and calculates the $\mu\_C(s)$ on each additional load.

$$\mu\_C(s) = \frac{1}{C}\sum_{i=1}^{n} C(s) \qquad (11)$$

5.2 Simulation Result

The simulation environment is based on [11]. The system consisted of 10 clients where each client produces 10 loops of requests and 3 object replica to serve the request. The mean service time of each service was 300 ms. Each client had an arrival mean of 100 ms. In every arrival, the clients overlapped their arrival time. The estimated simulation time was obtained by multiplying the number of clients and the mean service then the result was divided by the number of object replica. The expected simulation time in a single client processing 10 loops will be approximately 0.01 second to process the requests ((10 * 0.003) / 3). However, the latency of the transfer time varies as the count of client increases. The system also supported queuing of multiple requests.

The result from the on-demand replication scheme (ORS), least load and round robin (RR) are compared to the proposed least load algorithm. ORR is the improvement of RR which mechanizes also the dynamic replication scheme [15]. The total mean client request completion (CRC) time of the proposed fuzzy least load algorithm is approximately 0.37 seconds while the two other algorithms which are the ORR, LL, and RR have 0.43, 0.39 and 0.47. The waiting time of each request is minimized on using the proposed fuzzy least load algorithm because of determining the appropriate object. In addition, the loads from the replica are equally distributed for balancing the loads unlike in RR and ORR. The graph of in Figure 17 represents the CRC time of FLL, LL, ORR and RR with respect on the number of clients. At higher client loads, FLL outperforms the ORR and RR as indicated by lower CRC time of the system.

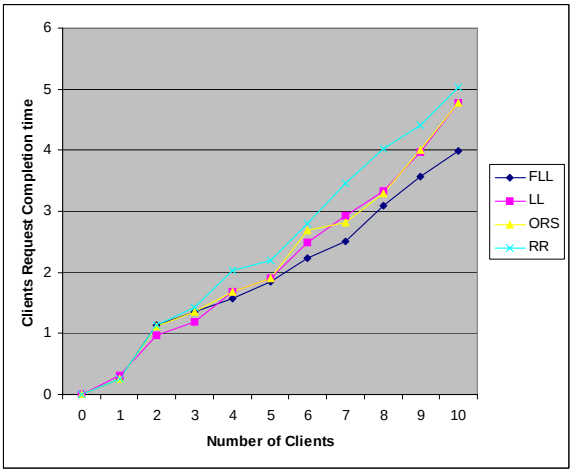


Fig. 17. Clients' request completion time on number of clients

6. Conclusions

This research presented the cooperation model for object group to implement the adaptive load balancing. The architecture of the distributed object group was discussed for the implementation of the proposed healthcare expert system. A single object group or service is managed by the group manager where it chooses the necessary objects from the replica server. The global load monitor and analyzer manage the flow of the adaptive scheme to implement the efficient distribution of loads. The replica manager was used to perform the replication of the replica. The adaptive load balancing is based on two algorithms which are the round robin and proposed fuzzy least load algorithm. The proposed fuzzy least load algorithm is used to process the fuzzy decision of distributing the loads. For the optimization of the load distribution, the fuzzy sets are trained from the previous data. This research study simulated the cooperation model for the object group implementing the efficient load distribution by using the proposed fuzzy least load algorithm. Evaluating the performance of the proposed algorithm, the total mean client request completion time of the system is measured and compared to other schemes [6, 7, 11].

References

[1]. Object Management Group: http://www.omg,org
[2]. Coulouris, G., Dollimore, J., and Kindberg, T.: Distributed Systems Concepts and Dessign. Addison Wesley, Third Edition (2001) 553 – 606
[3]. Defago, D. Felber, P. Schiper, A. : Optimization techniques for replicating CORBA objects. Proceedings of The Fourth International Workshop on Object-Oriented Real-Time Dependable Systems (1999) 2-8

[4]. Felber, P., Guerraoui, R., Schiper, A.: Evaluating CORBA portability: the case of an Object Group Service, Proceedings of The Second International Enterprise Distributed Object Computing Workshop (1998) 164-173
[5]. Zomaya, A. Y., The, Y-H.: Observations on Using Genetic Algorithms for Dynamic Load-Balancing. IEEE Transactions on Parallel and Distributed Systems, Vol. 12, No.9 (2001) 899-911
[6]. Othman, O., O'Ryan, C., and Schmidt, D. C.: The Design and Performance of an Adaptive CORBA Load Balancing Service, IEEE DS Online, Vol. 2, No. 4 (2001)
[7]. Balasubramanian, J., Schmidt, D.C., Dowdy, L. and Othman, O.: Evaluating the performance of middleware load balancing strategies, Eighth IEEE International Enterprise Distributed Object Computing Conference, (2004) 135-146
[8]. Cheung, L., and Kwok, Y.: A Fuzzy Load Balancing Service for Network Computing Based on Jini. LNCS Vol. 2150, Manchester, UK (2001) 183-190
[9]. Pope, A.: The CORBA Reference Guide. Addison-Wesley (1998) 25-33
[10]. Joo, S., Oh, S., Shin, C. and Hwang, J.: CORBA Based Real-Time Object-Group Platform in Distributed Computing Environment. ICCS (2003) 101-111
[11]. Mateo, R. M., Lee M., and Lee, J.: Dynamic Replication Scheme for Load Balancing the Distributed Object Group. In the Proceedings of the International Conference on Hybrid Information Technology 2006, Cheju Island (2006)
[12]. Dumitrescu, D., Lazzerini, B., and Jain, L. C.: Fuzzy Sets and Their Application to Clustering and Training. The CRC Press International Series on Computer Intelligence, CRC Press LLC (2000)
[13]. Ko, J., Gerardo, B. D., Lee, J. W., and Hwang, J.: The Information Search System Using Neural Network and Fuzzy Clustering Based on Mobile Agents. LNCS Vol. 3481, Singapore (2005) 205-214
[14]. Yi, S., Gerardo, B. D., Lee, Y. S., and Lee, J. W.: Intelligent Information Search Mechanism using Filtering and NFC based on Multi-agents inthe Distributed Environment. LNCS Vol. 3982, Glasgow, Scotland, U.K. (2006) 867-876
[15]. Halgamuge, S. K., and Glesner, M.: Neural Networks in Designing Fuzzy Systems for Real World Applications. Fuzzy Sets and Systems, 65 (1994) 1-12
[16]. Wai, R. J., and Chen, P. C.: Intelligent Tracking Control for Robot Manipulator Including Actuator Dynamics via TSK-type Fuzzy Neural Network. IEEE Trans. Fuzzy Systems Vol. 12 (2004) 552-560
[17]. Mitra, S., and Hayashi, Y.: Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework. IEEE Trans. Neural Networks, Vol. 11 (2000) 748-768
[18]. Pizzi, N. J., Pedrycz, W.: Fuzzy Set Theoretic Adjustment to Training Set Class Labels Using Robust Location Measures. IEEE
[19]. Kruse, R., Bolgelt, C., and Nauck, D.: Fuzzy Data Analysis: Challenges and Perspectives. In Proceedings of the 8th IEEE International Conference on Fuzzy Systems, IEEE Press, Piscataway, NJ, USA (1999)
[20]. Klose, A., Nürnberger, A., Nauck , D., and Kruse R.: Data Mining with Neuro-Fuzzy Models. Data Mining and Computational Intelligence, Springer-Verlag (2001) 1-36

Romeo Mark A. Mateo was born in Parañaque City, Philippines on February 3, 1983. He received his B.S. in Information Technology from West Visayas State University, Philippines in 2004. During 2005-2006, he worked at the Distributed Systems Laboratory (DSL), for research in distributed systems, mobile computing, data mining and artificial intelligence. Currently, he is completing his M.S. in Information and Telecommunications Engineering at the School of Electronic and Information Engineering in Kunsan National University, South Korea.

Insook Yoon received her B.S., and M.S., degrees in Information and Telecommunication from Kunsan National University in 1992 and 1999, respectively. Currently, she is a Ph.D. student at the School of Electronic and Information Engineering in Kunsan National University, Kunsan City, South Korea. Her research interests include distributed systems, database systems, data mining and computer networks.

Jaewan Lee received his B.S., M.S., and Ph.D. degrees in Computer Engineering from Chung-Ang University in 1984, 1987, and 1992, respectively. Currently, he is a professor at the School of Electronic and Information Engineering in Kunsan National University, Kunsan City, South Korea. His research interests include distributed systems, database systems, data mining and computer networks.