A Service Replication Scheme for Service Oriented Computing in Pervasive Environment

Sital Dash, Mridulina Nandi & Subhrabrata Choudhury

National Institute of Technology, Durgapur
E-mail : Sheetal.dash@gmail.com, mridulina.nandi@gmail.com, subhrabrata@gmail.com

Abstract – **In this paper we propose a service replication scheme for service oriented computing in pervasive environment .The scheme monitors the necessity of replication of each individual service and if a replication is required it determines the most suitable node at which the service is to be replicated. When there is a need for replication for a specific service , a parameter is evaluated for each node that represent the suitability of the node for replicating the service. This parameter is calculated as a weighted sum of essential parameters of a node such as remaining CPU slots, remaining battery power, and response time of each node for each specific service. The efficiency of the scheme is analyzed against existing replication schemes and it is found to perform better in terms of response time and resource utilization especially in a pervasive environment characterized by frequent topology variation and heterogeneity.**

Keywords – Service Replication; SOA ; Topology Variation; P2P .

I. INTRODUCTION

Service oriented computing [1],[2] provides a powerful mechanism to deliver business processes for its simplicity, reusability and flexibility (in new service composition). Service providers publishes services in the Universal Description, Discovery and Integration(UDDI)[3] and when that services are required to fulfill a task then service client discovers services. Thus Service-Oriented Computing is very attractive in pervasive computing where small handheld devices compute complex task in co-operation with each other. But it is very difficult to implement Service-Oriented Architecture (SOA) in pervasive [4] environment because of its centralized nature, frequent service unavailability resulting from poor channel conditions, power saving strategies and user mobility as it inherits the properties of client/server based architecture.

To combat the situation we proposed an agent based peer-to-peer SOA(P2P-SOA) in [5] and we developed an analytical framework that shows the performance P2P-SOA[6] is better than the client/server based SOA in terms of response time and service availability. Service replication is a good choice to achieve service availability and to provide expected response time as desired by clients. Robustness, high service availability and load balancing are achieved through service replication. If we replicate the service then expected search time and search space will be reduced significantly.

Replication strategies can be divided broadly into 3 types *ServerCentric* replication [7]-[9] where replication is done by the service provider, *ClientCentric* replication [10] where replication is done by the service consumer and *Path* replication[11],[12] where replicas are created in the nodes which are in between the paths of service provider and every service consumer. *Server-Centric* replication provides high efficiency in query solving but it is an overhead on the neighbors of the service provider. In the *ClientCentric* replication, Light-weight, Adaptive and System-neutral Replication protocol (LAR) [10] replicate files in service consumers. To solve query, *ClientCentric* replicas are also efficient but the authors show low performance in serving other service consumers. *Path* replication is effective in the sense of query solving and hit rate but it is an extreme overhead to create a replica between the service provider and every active clients of that service i.e. number of replica is equal to number of active clients.

RequestCentric scheme such as uniform, proportional and square-root replication are proposed in [17]. In uniform replication, the replication factor doesn't depend on the popularity of the services all items equally replicated. In proportional replication, replication factor is directly proportional with the popularity distribution of the item and again replication factor is proportional with the square root of the popularity distribution of the item in square root replication scheme. All the of the above *RequestCentric* scheme focuses on how much to be replicated i.e. the replication factors but the author don't specify the situation when replication is indeed needed and also don't consider the selection of the node in which replication will be done.

A distributed replication technique is proposed in [14] where the whole network is divided into zones. The author consider a zone head for every zone depending on the mobility of the peer, when replication is needed in certain zone then the service will be replicated in the zone head. The authors selected the zone head based on mobility factor but don't consider the computing power. If the zone head keeps on replicating frequently, it will be overloaded and unable to respond properly.

Efficient and Adaptive Decentralized replication algorithm (EAD) is proposed in [15] which select the query traffic hubs that receive many queries of a file and frequent requesters as replica nodes. But with strategy the replication algorithm will become topology dependent. But none of the above techniques consider the capacity of the node in term of computing power and battery power in which replica will be created.

In this paper, we propose a scheme which keeps on evaluating the necessity of replication and if so the replica is created on a node which maximizes the service availability at minimum possible response time. For selecting the node where the service replica should be created, we evaluate a quantitative parameter Q for all nodes in the network. Q is weighted sum of essential parameters of a node such as remaining CPU slots, remaining battery power, and response time provided by that node to the active clients of the service. Then the service will be replicated to the node which has highest Q value.

The paper is organized as follows. In section II, we give a brief idea of the proposed system model. We present the details of our proposed service replication scheme in section III. We compare our replication scheme with square root replication scheme and EAD replication scheme in section IV. We draw our conclusions in section V with a brief outline of the future scopes.

II. SYSTEM MODEL

Here we assume that every peer can act as clients for some services and also they act as service providing peer for some services and every peer have an agent program, the agents are the autonomous programs which can take its own decision. In the figure 2.1 we show the system model. An agent of each node communicates with agents of other nodes and forms a middleware which provides another level of abstraction.

These peers (agents) exchange some messages between them through which they are able to decide when service replication is actually necessary and how to choice the best candidate for replicate the service. We assume that one service can be provided by unique server. Each server in the network has service-table which contains the name of the services that is provided by a server. So initially there is no replication in the service-table of each server.

Some peer holds distributed UDDI [3] table which are basically service information of some nodes. In this way all these tables hold the service information of all UDDI table as each service is provided by the unique server.

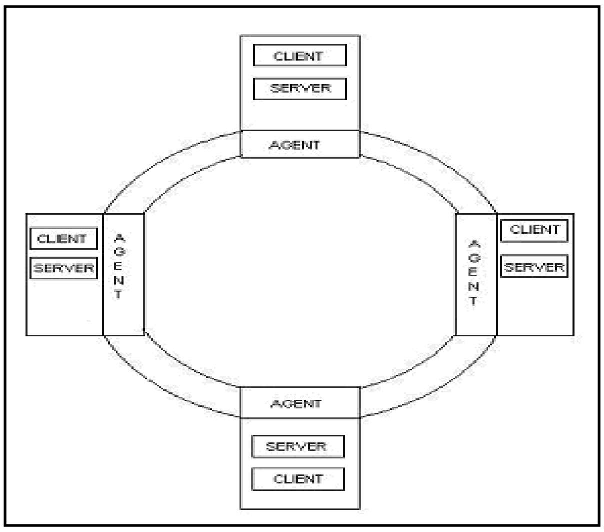

Fig. 2.1 : Proposed System Model

III. PROPOSED REPLICATION SCHEME

Service replication is an overhead to the network, so, prior to create the replicas we have to identify the situation when replication is really needed and after we decides to replicate the service we have to create the replicas in the best possible nodes. So our service replication scheme can be divided into two sections.

- When to replicate the service
- Where (in which node) to replicate the service

A. When to replicate the service:

The replica should be created under these following issues

- *Power Consumption and Mobility:* P2P-SOA in Pervasive environment every node suffers from power constrain. A server may unavailable due to switching off its battery either for less power or mobility of nodes in the network.

- *Response Time:* Response time is a crucial factor for both client and server. When server is not able to provide the expected response time due to overload and wants to reduce some load to get relief. On the other hand if the service clients are not getting the expected response time.

If we create the replica for Power Consumption issue then the server will send a message to the middleware informing that it will be unavailable near future. Let servers S_N are going to switch off within a period of time and then from service-table of S_N middleware gets the list of services provided by S_N. Then immediately middleware decides to replicate the services of that server to a suitable node. Middleware will not replicate all services provided by that S_N to a single node rather it replicates services of that server one by one. The service-table of the suitable node will be updated at that time as well as the distributed UDDI table will also be updated.

If middleware wants to create the replica for improving Response Time then it will not replicate all the services provided by the S_N rather middleware will replicate only those services one by one for which clients expect the response time is not meeting or replicate those services or portion of those services by replicating whom the middleware will able to reduce the load of that overloaded server. When the middleware wants to replicate the service, the prime issue to be consider is that which node will be the best candidate for service replication.

B. Where to replicate the service:

To select the best candidate for replication we have used a reactive approach. With this approach every node in the network will calculate a quality parameter *Q* for every services provided by the nodes in the network. *Q* is the weighted sum of all crucial parameters of a node such as remaining CPU slots (*CSre*) that reflects the load on the node, the remaining battery power (*BPre*), delay in term of response time (*RTavg*) provided to the active clients of the service. It takes care of load, delay and battery life time of a node. Emphasis on each of the parameters , weights can be set by each individual node independently or all nodes can choose same weights based on current scenario. After calculating Q parameter in each node for each service, it will be flooded among nodes by which middleware can get *Q* parameter values of all other nodes in the network.

- *Remaining CPU Slot:* This component reveal how much loaded a node is. If for a node the value of this component is very low means that the node is already overloaded and it is meaningless to make it more loaded by replicating another service. So its contribution on will be less as its respective value is less.

- *Remaining Battery Power:* Battery power is very crucial factor to consider when we selecting a node for replicating service. If we replicate a service to a node that have very less amount of power than after a few period of time that node will be switched off and replication is needed again, which is unnecessary overhead to a network. To avoid repeated replication it is desired to replicate a service to a node which have reasonable amount of battery power.

- *Response Time:* When a node calculate this parameter for a service it has to measure how much response time it should provide to the active clients of that service because it reflects the delay the active clients have to suffer if the node is selected for replication. If for a particular service average response time of a node from its active client set is less than others then the node will gate quick response than other nodes in the network and will be selected as best node. As the underlying routing protocol is reactive then the respective node which is calculating has to measure it through following steps.

1. Let peer $n1$ want to calculate average response time *RTavg* for all active clients of the service *j*. Let the active clients of service *j*.

$$^{client}Ser_j = \{c_1, c_2\} \qquad (3.1)$$

2. Peer n_1 sends a DSR route request packet to all member of the above active client set.
3. The receivers of route request packet then send a route reply packet to n_1 and with that packet the receiver i.e. the clients send a message to calculate the delay in every intermediate node.
4. When the route reply packet is arrived at each intermediate node they calculate the delay in the link from their transmission time and queue length and send it with the route reply packet.
5. When the route reply packets from every member of the set arrived at n_1, it is able to know the response time between $n1$ and the active clients.
6. From all those response times among active clients and node n_1, node n_1 calculate average response time. In this way all other nodes in the network calculate average response time (*RTavg*) for that particular service *j*.

C. *Sequential Steps for Selecting the Best Node for Replicating a Service :*

1. Replication can occur in two cases
   a) low Remaining CPU Slot
   b) low Battery Power.
2. If middleware decide for replication then every node in the network except the server node S_N calculate *Q* parameter value.
3. *Q* is the weighted sum of Remaining CPU Slot (*CSre)*, Remaining Battery Power (BP_{re}) and Average Response Time (RT_{avg}). *Q* value is calculated in that way

   $Q = \{W_C.(CS_{re}) + W_B.(BP_{re}) + W_R.(1 - RT_{avg} / RT_{max})\}$

   Where

   W_C is the weight of remaining cpu slot.

   W_B is the weight of remaining battery power.

   W_R is the weight of average response time.

   And

   $W_C + W_B + W_R = 1$
4. If replication occur for low remaining cpu slot then for service *j* of server S_N then there must be active client set for that service of that server and reactively all the nodes in the network calculate average response time (RT_{avg}) among active clients in the above average response time algorithm.
5. If replication occur for low battery power of server S_N then for all services of that server there may be active client set or not. If there exist active client set for a service then all the nodes in the network calculate average response time (RT_{avg}) otherwise this term will be zero.
6. The best node will be max *Q* paremeter value node as in best node remaining cpu slot and remaining battery power will be maximum than other nodes but response time will be minimum than other nodes. So (1 - RT_{avg} / RT_{max}) this term will be maximum for best node.
7. After calculating *Q* parameter value each node broadcast this value to other nodes in the network and middleware find out the max *Q* value holder node.
8. Finally middleware replicate the service to highest *Q* value holder node.

D. *Scalability:*

In our replication strategy when replication is necessary for any service then every node in the network will calculate the *Q* parameter for that service and agents are collecting the *Q* parameters for that service from every node through flooding. These scheme will perform well in small network with few number of nodes. But when the network becomes large, the collection process *Q* parameter will produce a huge amount of network traffic, latency and thus replication process will become no scalable. So to make it scalable we propose a clustering mechanism.

E. *Clustering:*

When the network becomes large we divide the network into *R* hops clusters. Each cluster will have a cluster head. We select the cluster head depending on the degree of connectivity with the neighbors and the link stability of it’s neighbors. When replication is necessary for a service then all member nodes of the network will calculate the *Q* parameter for that service and send it to their respective zone head. The zone head will then decide who will be the highest *Q* holder for that particular service and store that information. And then all the zonal head agents will communicate with each other and find out the max *Q* holder node.

IV. RESULT AND DISCUSSIONS

We considered an adhoc network with an area coverage of 500m × 500m and place 256 nodes in random position. Each node is having a transmission range of 50m .The remaining CPU slots, remaining battery power and link failure probability of each link are allocated to each node following uniform random distribution.

We calculate a parameter by taking the weighted sum of standard deviation of available CPU slots, standard deviation of battery power and link failure probability of various links and term the parameter as ***topology_variation***.

topology_variation = $W_{cpu} \cdot \sigma_{cpu} + W_{BP} \cdot \sigma_{BP} + W_f \cdot \sigma_f$

Where W_{cpu} is the weight of remaining CPU slot

W_{BP} is the weight of remaining battery power

W_f is the weight of link failure

σ_{cpu} is the standard deviation of CPU slot

σ_{BP} is the standard deviation of battery power

σ_f is the standard deviation of link failure probability of each node.

In fig 4.1 we plot the topology_variation against response time and compare it with EAD replication scheme and find out that our scheme will perform better against topology variation than EAD scheme. The differences of response time between our scheme and EAD scheme increases exponentially.

In fig 4.2 we compare our replication with EAD replication scheme in terms of response time and find out that our replication scheme will perform better than EAD scheme in case of response time.

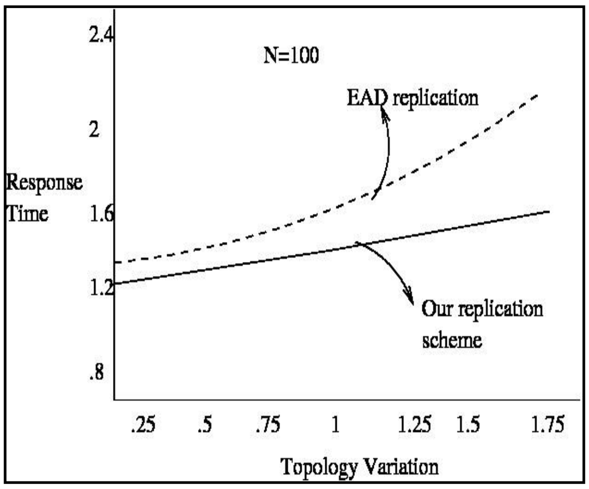

Fig. 4.1 : Comparison with EAD replication against topology variation

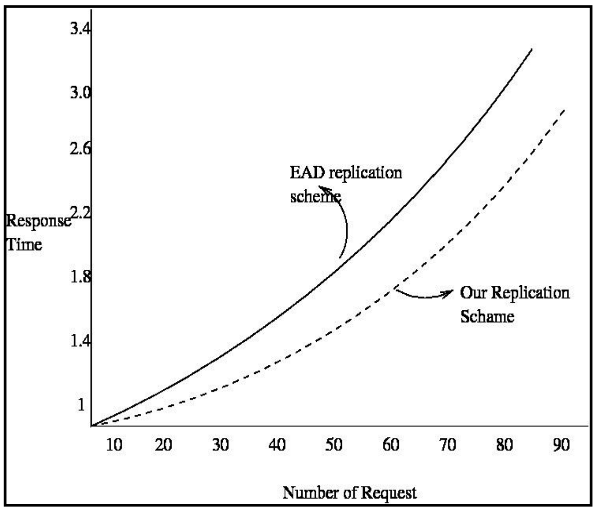

Fig. 4.2: Comparison with EAD replication against Response Time

In fig 4.3 we plot overhead bandwidth with number of nodes and compare it with EAD replication scheme. It gives better performance than EAD replication scheme for smaller network, but when number of nodes in the network increases i.e. network become large then performance of our scheme will not be better than EAD scheme because additional bandwidth required for sharing of information. However this additional bandwidth can be brought down by using the clustering scheme discussed above.

From the fig 4.4 , we observe that, square root replication gives better performance over our scheme in an wired network.

Again from fig 4.5 it is clear that overhead in memory and CPU allocations is more in square root replication because of extra replicas, as square root replication does not consider the necessity of replication. But in our scheme replica is created whenever it is necessary. Thus our scheme will perform better than square root replication in terms of resource utilization.

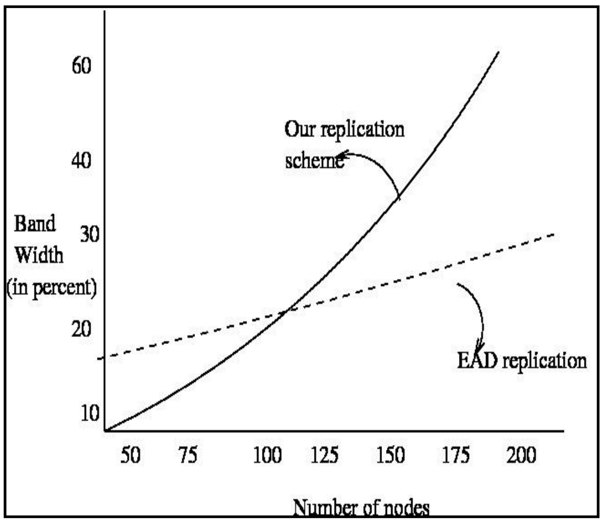

Fig. 4.3 : Comparison with EAD replication against Bandwidth

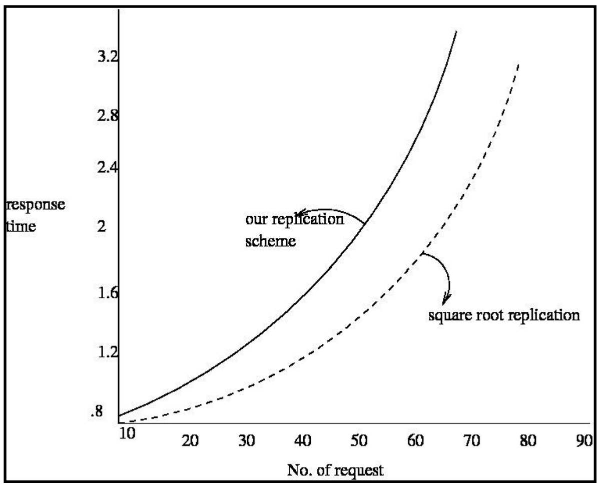

Fig. 4.4 : Comparison with Sqrt. replication against response time

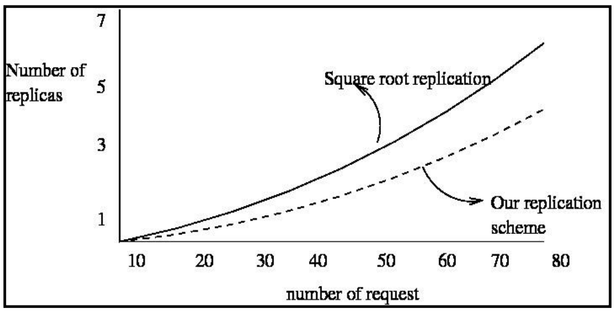

Fig. 4.5 : Comparison with Sqrt. replication against no. of replicas

Our replication algorithm gives better performance in terms of topology variation, response time, robustness against heterogeneity and also it optimizes the number of replicas. But the penalty is paid in term of bandwidth required for sharing information. This can again be reduced using clusters scheme.

V. CONCLUSION

In this paper, we developed a service replication scheme for service oriented computing in a pervasive environment. It evaluates the situation when a particular service is to be replicated and then determines the node which is most suitable for replicating the specific service. For doing so it evaluates a parameter that consider the essential parameters of a node such as remaining CPU slot, remaining battery power and response time for each individual service. It is found that the scheme gives better response time than all existing schemes under pervasive environment (characterized by large variation of node capability and frequent communication failure).

VI. REFERENCES

[1] Papazoglou , M.P. Traverso , P. Dustdar ,S. Leymann, F.,"Service-Oriented Computing : State of the Art and Research Challenges " ,IEEE Computer ,Vol. 40 pp38-45,Nov 2007.

[2] Papazoglou, M.P. Traverso , " Service –Oriented computing : concepts ,characteristics and directions ." In Proc. ,WISE 2003, 10-12 Dec 2003 ,pp 3-12.

[3] "Uddi data structure reference v1.0", available at http://uddi.org/pubs/DataStructure-V1.00-Published-20020628,June,2002.

[4] M. S. Nicklous U. Hansmann, L. Mark and Th. Stober. "Pervasive Computing-The Mobile World"-2nd ed. Springer professional series, London, 2003.

[5] S. Choudury, D. Mukharjee, P. Chakraborty, N. C. Debnath, and H. K. Lee. "An agent based framework for peer to peer distributed computing." In Proc. ISCA-CATA, pages 252–257, April 2008.

[6] Piyali De, Prasenjit Choudhury, and S. Choudhury. "A framework for performance analysis of client/server based SOA and P2P SOA". ICCNT-2010,April 2010.

[7] A. Rowstron and P. Druschel. "Storage management and caching in past, a large scale, persistent peer-to-peer storage utility. "In Proc. of SOSP, April 2001.

[8] T. Stading and et al. "Peer-to-peer caching schemes to address flash crowds." In Proc. of IPTPS, 2002.

[9] M. Theimer and M. Jones. "Overlook: Scalable name service on an overlay network." In Proc. of ICDCS, 2002.

[10] V. Gopalakrishnan , B. Silaghi, "Adaptive replication in peer-to-peer systems." In Proc. of ICDCS, 2004.

[11] M. Roussopoulos and M. Baker. CUP. "Controlled update propagation in peer to peer networks." In Proc. Of USENIX, 2003.

[12] L. Yin and G. Cao. DUP. "Dynamic-tree based update propagation in peer-to-peer networks." In Proc. of ICDE, 2005

[13] Edith Cohen and Scott Shenker, "Replication strategies in unstructured peer-to-peer networks." SIGCOMM02, pages 19–23, 200A.

[14] Ahmed, K. Yasumoto, N. Shibata, T. Kitani, and M. Ito. Dar:"Distributed adaptive service replication for manets." International Conference on Wireless and Mobile Computing, Networking and Communications, 2009.

[15] Haiying Shen." EAD: "An efficient and adaptive decentralized file replication algorithm in p2p files sharing systems." Eighth International Conference on Peer-to-Peer computing, pages 99–108, 2008.

◈◈◈