# CONSISTENCY MAINTAINING REPLICA IN WEB SERVICES

Regina Begam.M*

*Abstract* —
This paper presents, an ISS (Intelligence Surrogate System) is used to manage the replication of the web services. The middleware uses the Time Stamp Based (TSR) protocol to handle the synchronization operations to ensure that the replicas' states are consistent. A web service is replicated at several sites. Each replica consists of a proxy web service site (PWSS) and a web service site (WSS). The WSS is a conventional web service provider. It hosts the code and data that provide the functionality of the web services. The PWSS is residing between the clients and the WSS. It is responsible for ensuring the consistency of the replica. Clients interact with the PWSSs.
A client only needs to send its service request to one PWSS. The PWSSs are responsible for multicasting clients' requests to other replicas and returning results to the clients. To maintain the consistency of the WSSs' states, the PWSSs must ensure that all clients' requests are executed on the WSSs in the same order. The replicas form a group, called service group. ATU (Auto Time Updater) is responsible to manage when none of response occurs.

***Keywords***— **ISS (Intelligence Surrogate System), PWSS (Proxy Web Service Site), WSS (Web Service Site), ATU (Auto Time Updater), TSR (Time Stamp Based).**

* *Assistant Professor ,computer Science Department, SBM College of Enginnering,Dindigul-624005*

## I. INTRODUCTION

Webservice is a software system designed to support interoperable machine-to-machine interaction over a network. Web services are application components. Web services communicate using open protocols. Web services are self-contained components and self-describing XML is the basis for Web services.

A middleware that supports reliable web services using the active replication technique. It uses a Time-Stamp based protocol to maintain the consistency of the replicas' states. Service List, Service group, TSR [1] Scheduler and ATU [1] are the components of Middleware to execute the services. Inter link communication between client and WSS can be achieved by PWSS.ATU based time updating service for none of response occurs.

## II. MOTIVATIONS

Web services are self-contained, modular applications that can be located and invoked over the Internet. As more and more applications are built on web services, providing reliable web services is becoming an important issue.

The proposed system presents a middleware that transparently

regisaleem@gmail.com

supports reliable web services using the active replication technique.[4,12]

The middleware is responsible for maintaining the consistency of replicas' states. It uses a time stamp based protocol to maintain the consistency of the replicas' states. {wait for the ordering messages from the sequencer before executing clients' requests, for some applications, it is possible to devise a scheme that achieves better performance than OAR (Optimistic Active Replication Protocol)[6] Compared with the optimistic active replication protocol and replication schemes based on the group communication primitives, the protocol used in this system reduces the replication overhead for a class of applications.

## III. THE SYSTEM

### *A. An Overview*

The following assumptions are made about the system.

A web service is replicated at several sites. Each replica consists of a proxy web service site (PWSS) and a web service site (WSS). The WSS is a conventional web service provider. It hosts the code and data that provide the functionality of the web services. The PWSS is residing between the clients and the WSS. It is responsible for ensuring the consistency of the replica. Clients interact with the PWSSs. A client only needs to send its service request to one PWSS. The PWSSs are responsible for multicasting clients' requests to other replicas and returning results to the clients. To maintain the consistency of the WSSs' states, the PWSSs must ensure that all clients' requests are executed on the WSSs in the same order. The replicas form a group, called service group.

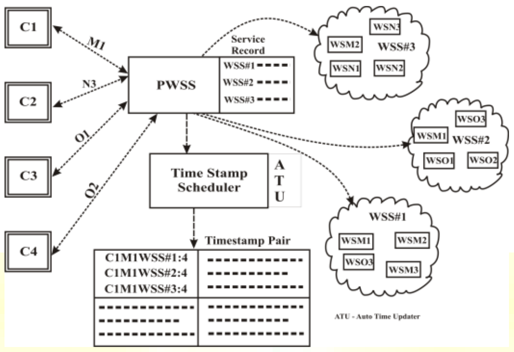


Fig.1 Using TSR Protocol

TSR Protocol the sequencer will be totally avoided and ordering messages will be eliminated. PWSS is the proxy sites between web service and clients. PWSS maintains service records and service group. Service Records are the key value pairs of service details. Service group consists the request information against WSS to list the update features. Timestamp scheduler is the separate component of PWSS, to schedule Timestamp pair c1m1wss#:4. ATU is a Subcomponent of TSR to listen response timing and update Timestamp pair(TSP) time, when none of response is occurred

## B. Design Details

*1)Service Group Formation:* Clients are made service requests to PWSS. PWSS consists the sub component TSR protocol, ATU is the sub component of TSR. Input goes to Web Service Site by PWSS for clients. Web service sites are the application servers and they have run time environment of software tools to execute the services. Service searching, service finding, executing are the tasks of web service site.

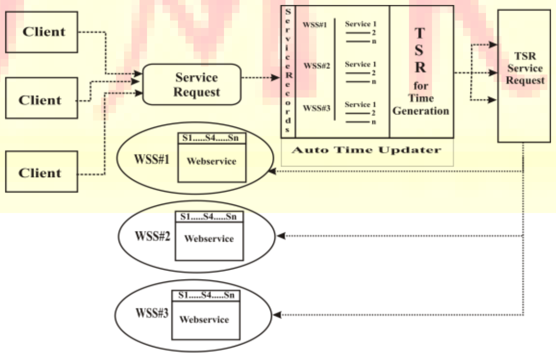


Fig.2 Service Group Formation

*2)Service Group Algorithm*: User made service requests to PWSS, the requests are made demand basis service list. If the list is found then the target web services established.

TABLE I

| | |
|---|---|
| 1. | User made service requests |
| 2. | Demand basis service list verdict |
| 3. | Target web services establishment<br>C1M1WSS#1:5<br>C1 : user name<br>M1: Service name<br>WSS: The name of web service site<br>5 indicates random time |

*3)Time Stamp Scheduler Algorithm:* Based on WSDL (Web Service Description Language) web service engine finds the web service. Various web service sites have different services and analyzer (Run Time Environment) executes the appropriate web service hosted in WSS. Result will be returned to PWSS, in same time TSR and ATU manage the clock time for the response timings. Based on service group the result will be collected and arranged as one collection then it will be returned to clients. RM is the process Request Manager and RN is the process of Request Next tasks.

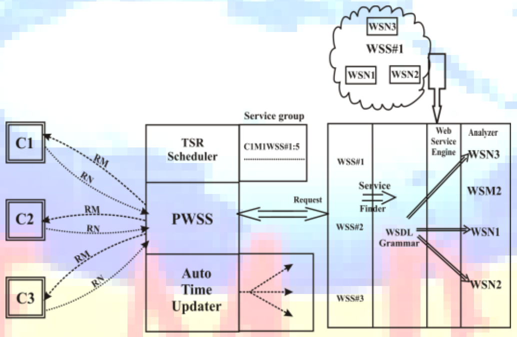


Fig.3 Task Flow of webservice with PWSS

PWSS decide which services are to be connected. Based on the service group format random timing is chosen. Multiple demands will be started with the timing fashion.

TABLE II

| | |
|---|---|
| 1. | Deciding which services are to Be connected |
| 2. | Based on service Group format random timing chosen |
| 3. | Multiple Demands will be started with the timing fashion.<br>C1M1WSS#1:5 |

*4)ATU Algorithm:* ATU listening and responding contents from various web services. If the time is overflow the scheduler will automatically increase the timer to new time. If none of results the time will be updated automatically. Rerequest is started.

TABLE III

| | |
|---|---|
| 1. | Listening Responding contents from various web services. |
| 2. | Time overflow adjustment new scheduler. |
| 3. | Time Updating for none of results. |
| 4. | Request is started.<br>C1M1WSS#1:5:f<br>C1M1WSS#2:5:nf<br>C1M1WSS#2:7:f |

C. The Implementation of The System

Whole processes are arranged into three phases. Phase I is the web service development and the hosting tasks. Web service consists the business logic of the application, One or more web services are hosted in different web service sites. Phase II is the process of PWSS and its sub component which acts as middleware between client application and WSS. Phase III is the client application which has three components Service Manager, Request Manager and Message Consumer.

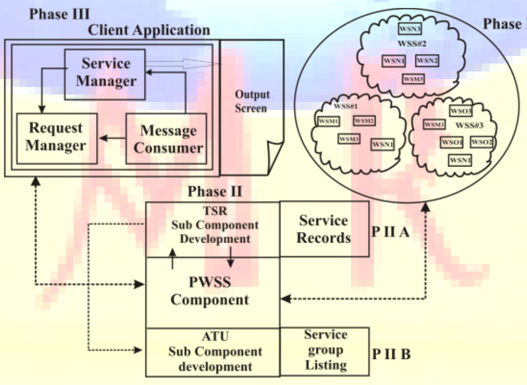


Fig.4 Complete Phases Flow Logic

Service request is the origin of the client side interaction after authentication. GUI based interaction pages shows the user to get the service choice and start the request manager apps with message consumer listener.

Proxy web service site is the middleware and the main component of MW. TSR (Time Stamp Based Replication) sub component with ATU (Auto Time updater) instead of OAR (Optimistic Active Replication Protocol).[6] These are bundled together to execute the service requests in web service sites. PWSS has the service records of WSS for the fast execution when the SR is received.

Time stamp based scheduler creates the schedule for the service requests automatically without any separate invocation. The format of schedule can be cnsnwss:rt (client name ;service name; webservicesitename ;random time).

Web service provider is an application server and web service is the executable component to achieve the business logic of application with communicative tasks such as WSDL (Web Services Description Language). PWSS connects web service provider to get the results of service request.

ATU is the sub component of PWSS which is used to manage time defined by TSR and update the time if the none of response occurs

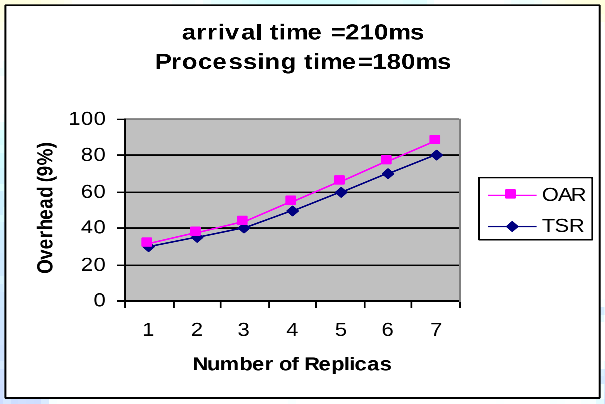
PWSS communicates with different WSS to execute the web services located in different servers. The service group is managed by PWSS to get back the results from the service provider. After the task is finished it will be destroyed and next SG is started for next service request.
PWSS collects the various responses within the allotted time and the response will be returned back to client. The returned results are prepared by web services. Rerequest is the process if client is not satisfied with the returned results then rerequest can be started and PWSS does the rerequest against web services.

## IV. PERFORMANCE

Experiments were carried out on machines to measure the overheads of the PWSS when a service group is deployed over a WAN. In the experiments, the replicas are connect by Ethernet. To simulate that the replicas are connected by a WAN, each message exchanged amongst the PWSSs is held in a buffer for 120ms before it is processed (i.e. the WAN has a one way delay of 120ms). Overhead is defined as $(tn - t1)$ $t1$where $tn$ is the service response time when the service group has $n$ replicas and $t1$ is the service response time when no replication is used.

In the following discussion, *arrival rate* means the inverse of the interval between the clients' requests arrive at the system while *processing rate* is the inverse of the duration to process a client's request by the WSS. The arrival rate and the processing rate are both simulated to follow an exponential distribution. The generated clients' requests are randomly distributed to the replicas. It is assumed that the arrival rate is less than the processing rate. That is, the system has sufficient processing capacity to handle the clients' requests.

The first experiment compares the performance of the *OAR* and the *TSR* when they are used by the PWSS to ensure the consistency of the replicas. In the *OAR* based implementation, the clients are modified to send their requests to all the PWSSs instead of a single PWSS. Also, the clients will collect replies from all the PWSSs for each of their requests. The experiments assume that it is rare that different clients send requests to the PWSSs simultaneously. Figure 2(a) and Figure2(b) show the performance of the system under different mean arrival interval and processing time. It can be seen that the *TSR* protocol outperforms the *OAR* protocol in both cases. It also can be seen that, when the mean processing time increases, the overhead decreases due to the reduced synchronization cost (i.e. the cost for ordering the clients' requests amongst the replicas) for per unit of computation. Since the clients hardly send requests to the PWSSs simultaneously, the number of requests that are executed in wrong order when using the *TSR* protocol does not increase significantly as the number of replicas increases. As a result, the costs for undoing and re-executing the requests that were executed in the wrong order do not increase significantly. Thus, the synchronization costs in running the *TSR* protocol dominate the overheads. For both protocols, as more replicas are added to the systems, it takes longer to run the protocols amongst all the replicas in the system. As *OAR* and *TSR* both require one round of multicasting and one round of acknowledging/sending response, the costs of running the two protocols increase at roughly the same rate.Hence, it can be seen that the differences in the overheads of the protocols are about 10% and 20% in Figure 2(a) and 2(b) respectively.



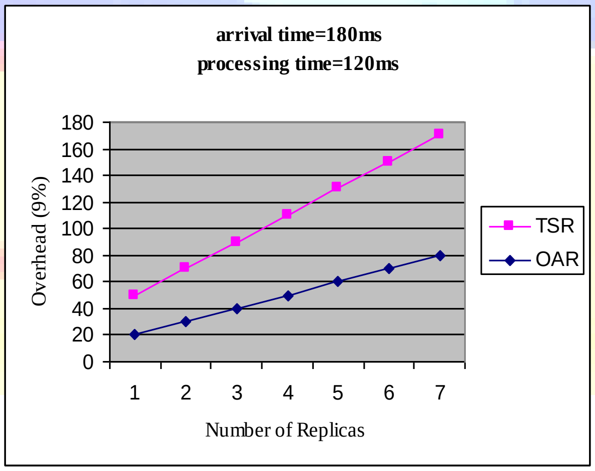


Fig.5 Comparing the Performance of TSR and OAR

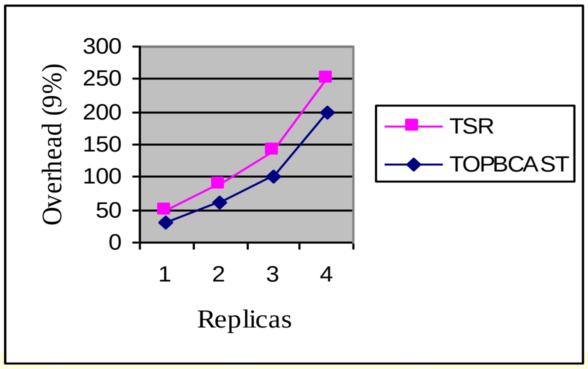


Fig.6 Comparison of TSR and TOPBCAST

The middleware uses a group communication protocol, i.e. the *TOPBCAST* protocol, to ensure the consistency of the replicas' states. Experiments were carried out to compare the performance of the middleware with the middleware discussed in this paper. Figure 3 shows the performance of the two middleware. From the figure, it can be seen that the two approaches have similar overheads when there are two replicas in the system. However, when there are more than two replicas in the system, the overhead of the middleware in this paper is significantly lower than the one.

## V. RELATED WORK

The proxy can be eliminated and the embedded proxy based client will be developed. It is the process of define cost less application and eliminating network congestion to completely depends proxy. The EP (embedded proxy) is the component of client and finds available services with the management of TSR with ATU sub component. By using this the application can get real benefit that numerous clients can be started without the overload of existing proxy. It means each one proxy will be started and managed the client application stability with data consistency.

Client starts service request. ISS catches service request. The chain based requests are catched. There are two types of chain based requests. a) Continuous service request: It will search the entire request in ISS. b) Discontinuous service request: It will search the request in ISS, if it is discover it will terminate.

Our experiment shows that the middleware based on the *TSR* protocol outperforms the one based on the *TOPBCAST* protocol. Clustering has been used to achieve scalability and availability in enterprise applications. Machines in cluster architecture are normally located close to each other and are connected by either proprietary hardware link/bus-interconnection or LAN-based interconnects. A load balancer distributes the clients' requests to the machines in the cluster. It is pointed out that the (i) network latency is one of the factors that affect an application's response time, and, (ii) one way to deal with the impact of latency is to locate web application geographically closer to users. Therefore, instead of hosting an application at a single site, the application might be replicated at several locations that are geographically far apart from each other. As a server cluster is normally deployed at a single site, the system proposed in this paper can be used to complement the cluster architecture for web applications. The system in this paper can be used to ensure that the requests for executing operations are delivered to the replicas at different geographical locations in the same order. At each location, a server cluster can be used to execute the received requests. As discussed earlier, for each operation, in order to guarantee the execution order, the requests for carrying out the operation have to be executed one by one. However, the requests for different operations can be carried out concurrently by the server cluster. Thus, deploying a server cluster at each replication site can still improve the overall performance of the site.

The Proposed schemes for executing the multicast messages optimistically. Like the *OAR* protocol, their schemes use a sequencer to (a) determine the order in which the multicast messages should be executed, and, (b) multicast the sequence numbers to all the replicas. However, they allow the multicast messages to be tentatively processed before receiving the sequence numbers from the sequencer. Compared to the schemes in [7] and [3], the *TSR* protocol does not use a sequencer to order the multicast messages. Thus, the *TSR* protocol does not concentrate the communication generated by multicasting the sequence numbers at any replica. Instead, the communication load is shared amongst all the replicas.

## VI. CONCLUSION

Active replication method in web service is three layer architecture. Layers are Service Request Application, PWSS (Proxy web service Site), WSS (Web service Site) – Application server. The system makes the communication between three layers through TSR protocol for the Web service consistency. .Net technologies are used to develop the layers to implement the cross platform networks.

The proxy can be eliminated and the embedded proxy based client will be developed. It means each one proxy will be started and managed the client application stability with data consistency.

The system discussed in this paper supports reliable web services based on active replication. As the system is designed as a middleware, it allows existing web services to be replicated at different sites without modifying the existing code. This allows web services to be replicated easily. The system uses a timestamp based active replication protocol.

## REFERENCES

[1] Ye,X., Providing Reliable Webservices through Active Replication, 6th IEEE/ACIS international Conference on Computer and Information Science,2007
[2] Ye, X., & shen, Y., A middleware for replicated Web services,Proceedings of 2005 IEEE International Conference on Web Services, pp. 631-638, 2005.
[3] J. Salas, F. P'erez-Sorrosal, M. Patino-Martnez, R. Jimenez- Peris, WS-Replication: A Framework for Highly Available web services, Proc. Of the 15th Intl. conf. on world Wide Web, pp.357-366,2006
[4] Microsoft ACE Team (2003), Performance Testing Microsoft .NET web Applications, Microsoft press
[5] Fred B.Schneider: Implementing Fault-Tolerant Services using the state machine Approach: A Tutorial. ACM Comput. Surv.22(4):299-319,1990
[6] P.Felber,A.Schiper,Optimistic active replication, Proc. Of 21st International Conference on Distributed Computing Systems, pp333 – 341, 2001
[7] A.Sousa, J.Pereira, F.Moura, R.Oliveria, Optimistic Total Order in Wide Area Networks, Proc.21st IEEE Symp. On Reliable Distributed Systems,190-199,2002.[8]L.Lamport,Time, clocks, and the Ordering of Events in a Distributed System. CACM, Vol.21, No. 7, pp558-565,1978
[8] K.P.Birman, Building Secure and Reliable Network Applications, Manning Publications Co., 1996
[9] Building XML Web Services For the Microsoft .NET platform, Microsoft press, Scott Short
[10] Visual Basic .NET Programming, Black Book, dreamtech
[11] Visual Studio .NET Programming, Mridula Parihar, NIIT , Yesh Singhal, Nitin Pandey.