International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org Volume 2 Issue 4 ǁ April. 2013 ǁ PP.07-13

A New Relation in SOA Model for Composite Web Service Atomic Transaction

Ezzat A. Korany[1], Fatma S. Gad El-Rab[2], Mahmoud M. El-Khouly[3]

[1]*(Department of information technology, Institute of Gradate Study and Research/ Alexandria University, Egypt)*
Department of Information Technology, Faculty of Computers & Information/ [2,3]*(Helwan University, Egypt)*

ABSTRACT: *The SOA (service oriented architecture) model of web-service could not grantee successful transaction between client and providers. Therefore client creates coordinator to handle transaction cases using standard protocol like business transaction protocol (BTP) and WS-Transactions. Mainly these protocols use ACID (Atomicity, Consistency, Isolation, Durability) properties and 2PC (2 Phase Commit) to handle transactions. These lead to problems in performance and lock resources for long time. This paper introduces new SOA model to decrease locking time for provider's resource and it compares a proposed model with a traditional model using BPEL (Business Process Execution language). The comparison results illustrate that proposed model is more efficient than traditional model in decreasing locking time for providers' resources.*

Keywords: *SOA, Web Services Interoperability, Composite Web Services, Three-Dimensional Modeling Composite Services, transaction web service*

I. INTRODUCTION

Composite web service transactions exist to ensure that all parts of a particular business operation are properly recorded. If any single part fails, it leads whole transaction to fail in order to maintain data consistency [1]. Web service can't manage transactions to handle this problem in composite web services to ensure data consistency because of its long running and loosely coupled nature [2]. Many protocols have been developed for (composite) web services transactions; WS-Transactions and OASIS business transaction protocol (BTP). They are mainly based on the database transaction models such as ACID properties and extended/advanced transaction models. ACID properties had been implemented using various commit protocols such as two-phase commit (2PC) protocol. Though ACID properties and 2PC protocols are useful in ensuring data consistency and correctness of transactions but they result serious performance problems in strict atomicity and isolation policy. ACID properties are useful for those web services which demand strict atomicity and consistency. However, they are inappropriate for long running business activities.WS-Transaction specification uses extended transaction models for business activities. Similarly, OASIS BTP uses an extended transaction model for long running tasks called cohesions. Extended transaction models mainly relax the strict atomicity and isolation policy of ACID properties such that intermediate results of active transactions are visible to other transactions. These models allow component transactions of a root transaction to commit unilaterally irrespective of the commitment of their Sib-ling transactions [3,4].

I.1 Background

Mikel [5], build algorithm that guarantees that eventually all the correct processes agree. Weihai [6], introduced the well known 2 PC optimization presumed commit and presumed abort and present an improved 2PC that is suitable for web services based application. It depends on time out to detect crash of providers. Also providers detect crashing of coordinator by time out this lead to stretch locking time of resources providers. Coordinator force write commit log record before send commit to provide also lead to stretch locking time of resources providers with time of force write operation and provider may crash after commit this cause data inconsistency. Boualem [7], proposed protocol compatibility and similarity in web services by developing a complete CASE tool supporting the web-service life cycle and choose to model a service e business protocol (protocol for short) as a non deterministic finite state machine. They did not take into account detecting crash of provider immediately. Muhammad [8], proposed a new commit protocol called TCP4CWS (transaction commit protocol for composite web services) which aims to improve the performance in committing a composite web service transaction. First writes its decision to a persistent storage before they can send a message. Thus the protocol is suspended until the forced write operation is completed this lead to performance issue and delay time for commit providers resources. They uses the processing time of a component web service transaction not execution time this lead to not accuracy time of process. That protocol does not take into account the crashing of client or one of providers during the execution these lead to increase locking time of resources and does not take into account the network latency time in calculates processing time of composite web service. Wenbing [9], presented mechanisms for a distributed transaction of a web services atomic transaction framework. The main novelty of them design is the minimized runtime overhead and the increased failure resiliency of distributed commit. The core mechanisms include a piggybacking mechanism, which limits the way of faulty coordinator replica can do to cause confusion among correct participants and a voting mechanism. These mechanisms did not take into account detecting crash of web service or client immediately so it waits to end process to explore failures. Changai [10], grouped the requirements with respect to ACID and adding deadlines to transactions. Comparison applied to BTP,WS-AT,WS-BA,BPEL and WS-CDL. The most of them depend on 2PC with relaxation model. Deadline of transaction based on business logic like time of reservation process. None of them takes into account crashing of client or dedicate crashing immediately. Mohammad [11], proposed a nonblocking schedule mechanism to use prior to the actual service invocations. It depended on deadlines of service providers to determine transaction time that mechanisms did not take into account crashing of providers during service provider time this lead to lock another provider to finish dead line at the first and take final decision (commit, abort). Pawel [12], presented a distributed commit protocol for supporting a wide variety of applications depends on 3PC and support network partitioning and multiple node failures. It commits another process to not fail, this leads to data inconsistency. That protocol depends on infinite transaction timeout to recover failed process to achieve full consistency this lead to increase looking recourses of data consistency. Wenbing [13], implemented new reservation protocol based on extended transaction, and classic transaction. The application has full control over the reservation and how long the resource is reserved, whereas in the two-phase commit protocol the locking of a resource is internal to the database system and is transparent to the application. The reservation of a resource is executed as a traditional ACID transaction and updating immediately of the state of the resource that is under control. That protocol does not take into account the crashing of client or one of providers during supervision and time of reservation process which leads to increase locking time of resources. Hossein [14], proposed a mechanism to SOA for recovering failure by distance measure which significantly reduces search space of hybrid search algorithm and results in near optimal solutions for composite web service, they ask user nonfunctional properties. Their method is not optimal but it is near optimal with significant reduced search space. It is a good solution to take into account network latency and how it effect in commit time and choose the best recovery process based this cost but it cannot dedicate crashing immediately.

II. PROPOSED MODEL

A proposed Model is improvement version for our SOA model (E.Korany [15]).

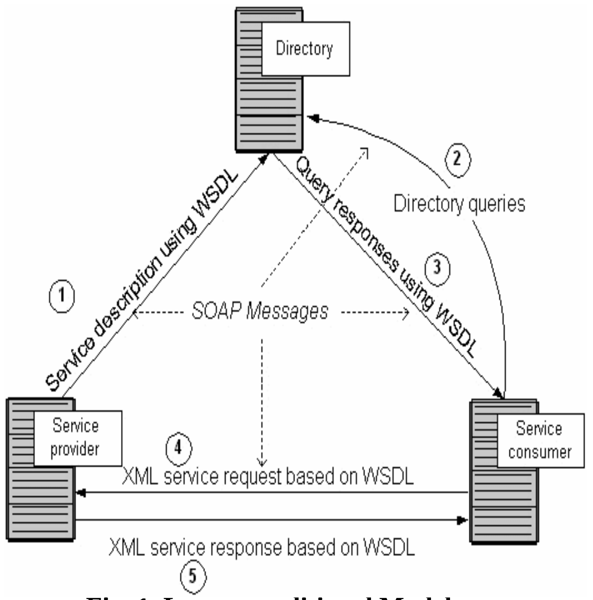
Directory
2
Directory queries
Service description using WSDL
1
Query responses using WSDL
3
SOAP Messages
Service provider
Service consumer
4
XML service request based on WSDL
XML service response based on WSDL
5

Fig. 1. Layout traditional Model

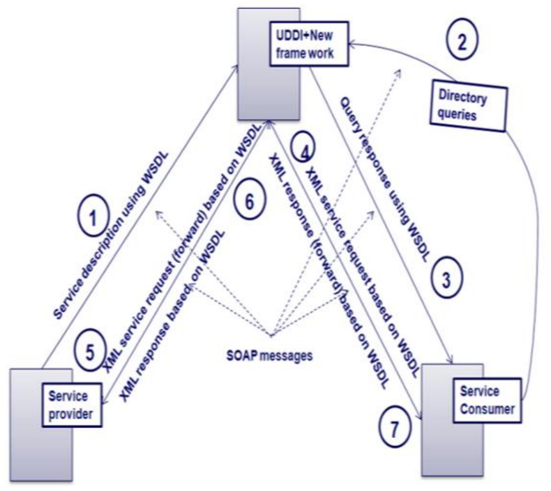
UDDI+New frame work
2
Directory queries
1
Service description using WSDL
XML service request (forward) based on WSDL
XML response based on WSDL
6
4
XML response (forward) based on WSDL
XML service request based on WSDL
Query response using WSDL
3
5
SOAP messages
Service provider
7
Service Consumer

Fig. 2. Layout proposed model [15]

Figure 1 illustrates traditional model of web service architecture, and figure 2 illustrates our previous model [15]. The first three steps are the same in both models. However, our improvement done in steps 4 to 7 as follows:

Step 4: Client send XML sheet to UDDI included data about composite web service (name, time, methods, parameters, WSDL URL).

Step 5: UDDI contains new framework to handle transaction between client and providers, UDDI receives request from client and forwards it to providers.
Step 6: Each provider should response to UDDI at the specific time (web service method time) in case of successful transaction. Step 7: UDDI collects response message from providers and compose it in one SOAP message to client in case of successful transaction. It detects error by error detection system for providers and client.

Participant's rules:
•**Client**'s rules: As mentioned in step 4.
•**Providers**' rules: Provider creates XML time sheet with the same name of service contains method's name and time. This time is actual time of execution to finish this method. This time sheet will be used later for handling transaction by coordinator. A proposed model handles transaction in a composite web service as an atomicity transaction (All or none). A proposed model needs provider to create three action methods (commit, abort, recover) per a web service method at the same class of a web service method. Action methods name should be the same name of web service method +name of action (create, abort, recover, commit) such that coordinator can call it. At first time coordinator calls provider (call web service method).

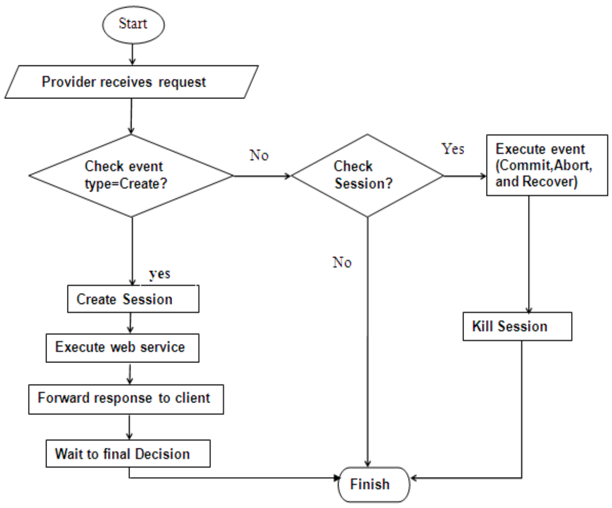

Fig. 3. Provider internal work flow

Figure 3 illustrates work flow within provider after receives the NUDDI request. In the proposed model the provider has to do two main steps (a) check event type, and (b) check session.
•**NUDDI**'s rules: As mentioned in step 5&7. NUDDI handles transactions in three cases (Normal cases, Failure of Consumer, Failure of at least one provider).
1.NUDDI rules with client: It receives SOAP request message from consumer, starting an error detection system for client, Extracting data from SOAP message, Calculating max time per request.

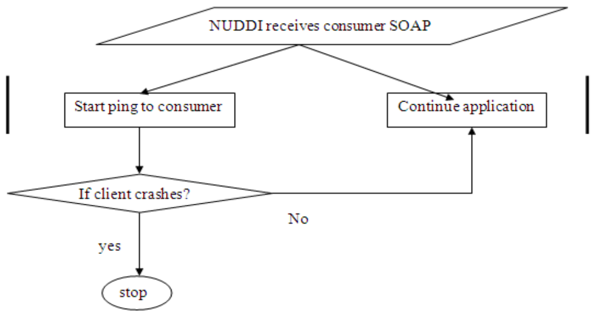

Fig. 4. Error detection system of client

Figure 4 illustrates how client Error detection system acts. A coordinator receives request from client and starts listening to client immediately

- **Normal case:-**No crash happens to client starting from send request to NUDDI to receive response.
- **Crashing case:-**Consumer crashes during execution of provider. This causes lost time to provider. Provider locks resources but no avail (no one will receive this response). Therefore new mechanism starts ping to client to detect crash of consumer immediately. NUDDI tells provider abort transaction

immediately also to free up resources.

2.NUDDI rules with providers: Sending SOAP request to each provider and Starting an error detection system per provider. NUDDI get time per ws. If NUDDI has three web service (ws1, ws2, ws3) nodes in client SOAP message. It has three web service times. Time is per web service method (wst1, wst2, wst3). NUDDI extract maximum time of these web services methods per SOAP request.

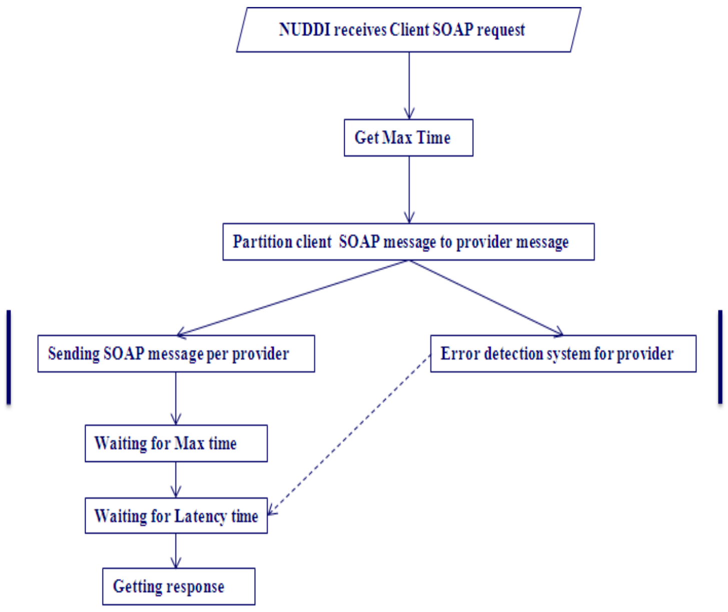
NUDDI receives Client SOAP request
Get Max Time
Partition client SOAP message to provider message
Sending SOAP message per provider
Error detection system for provider
Waiting for Max time
Waiting for Latency time
Getting response

Fig. 5. NUDDI sends SOAP request to each provider

Figure 5 illustrates NUDDI process sends SOAP request to each provider. NUDDI send SOAP to provider and wait for max time +latency to can get response.

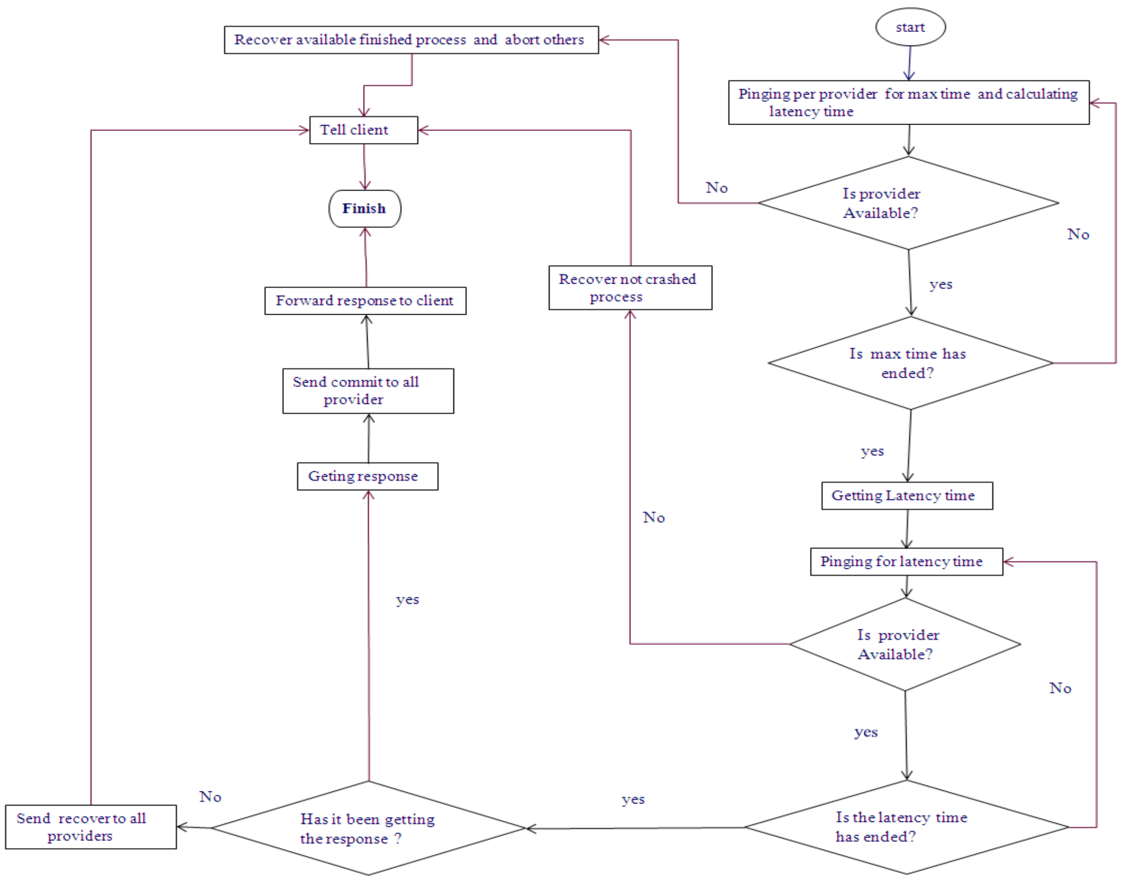
start
Pinging per provider for max time and calculating latency time
Is provider Available?
No
Recover available finished process and abort others
Tell client
Finish
No
yes
Is max time has ended?
yes
Getting Latency time
Pinging for latency time
Is provider Available?
No
Recover not crashed process
yes
No
Is the latency time has ended?
yes
Has it been getting the response ?
yes
Geting response
Send commit to all provider
Forward response to client
No
Send recover to all providers

Figure 6. Error detection system of provide

Figure 6 illustrates provider error detection system. This second part of error detection system. It used mainly to handle transaction for all providers to grantee all providers are not network crashing and can respond at exactly time. Error detection system supposes time for responding (WSR) per web service provider is measuring as actual time for web service method (WST) +network time (latency) (WSN). This time was calculated dynamically.

WSR=WSN+WST (1)

- **Normal cases:** In normal case NUDDI send final decision (commit) to all providers. Provider's response OK commit. NUDDI tells provider OK commit.
- **Crashing cases**: crashing may be in participants (client, coordinator, and provider) or network. In a composite web service atomic transaction. Client wishes getting result from all providers. If coordinator can't get response from at least one provider, coordinator has to be aborting all transaction.

III. RESULTS AND DISCUSSION

***PC properties used:*-Hardware**: - Intel(r) core(TM)2 duo CPU 5870@ 2 ghz,1995 MHz, 1. 99 GB).
Software used: - Windows xp sp3, Java: Jdk160_24, Oracle SOA suite 11g, Jdeveloper studio 11. 1. 1. 5 ,mysql 5. 1
Case Study:-Travel agency system. Client does reservation (hotel, car, and airline). Client needs atomicity transaction (All or none). Traditional model created using BPEL tools in oracle SOA suite. Client and providers opposed to crash (Network or per Se or both of them). Crashing event may be before starting transaction or within transaction. Proposed model detects crashing before and after stating transaction for client and providers but BPEL detects crashing before starting transaction for providers only note care to client.

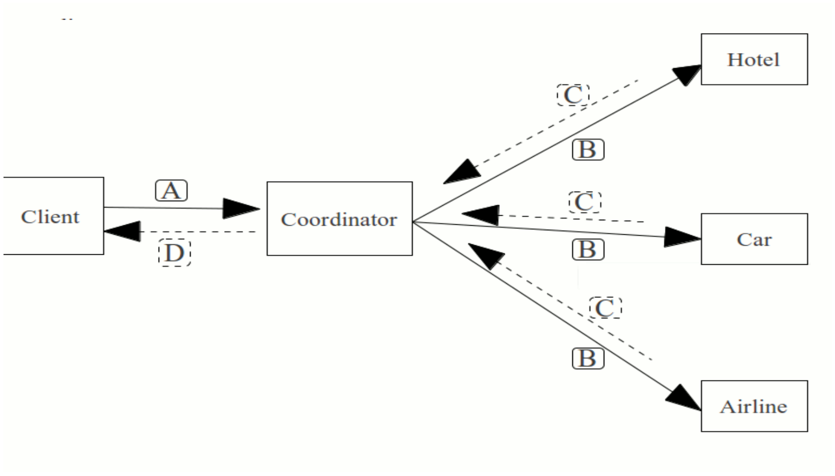

Figure 7 illustrates 4 stages of crashing as follows

A. Client sent request to coordinator but coordinator not forwards request to providers
B. Coordinator forwards request to providers and before coordinator receives response from providers
C. Coordinator receives response from provider but before forward it to client
D. Coordinator forward response to client or after client receives response from coordinator

Time calculated by millisecond				
Providers	**BPEL**	**New Model**		
		A	**B**	**C**
Airline	1099	0	355	455
Car	1099	0	286	477
Hotel	1339	0	352	336

Table 1 illustrates Locked time of provider resources in client crashing cases

A. ***Client crashes before coordinator forwards request to providers: -*** BPEL lock resources for many millisecond but proposed model does not consume any time because it can`t start transaction.

B. ***Client crashes after coordinator forwards request to providers and before coordinator receives***

response from providers:-proposed model check progress for each provider if it finishes execution, Coordinator sends recover; else coordinator sends abort to decrease locking time of row in database. Proposed model less than BPEL in locking time of resources because it has error detection system for client, but BPEL no so BPEL lock resource to finish execution.

C. ***Client crashes after coordinator receives response from provider but before forward it to client: -*** Coordinator sends recover to all providers to release row. But in BPEL not care with this crash.BPEL can`t feel if client crashed or no but in the proposed model can handle this issue and sends recover immediately.

Notes:-

D. ***Client crashes within coordinator forward response to client or after client receives response from coordinator: -*** *Here providers get final decision from coordinator so provider release resources but client can`t gets result.*

Table 2 Locked time of provider resources in provider crashing case

Time calculated by millisecond			
Provider	**BPEL**	**New Model**	
		B	**C**
Airline	30000	455	355
Car	30000	-	-
Hotel	30000	336	352

Table 2 illustrates Locked time of provider resources in ***crashing one of providers:***

A. ***Provider crashes before coordinator forwards request to providers: - Both of system (BPEL and proposed*** model can`t start transaction.

B. ***Provider crashing after receive request from coordinator***. Crashing may be in network (ping time out or per Se). Proposed model has error detection system to listen to providers if provider response ping time out, coordinator check time per another providers and sends abort or recovers according to time to another providers. BPEL has not error detection system so it can't send final decision to another process correctly. In BPEL if crashing before connect to database no response from BPEL else data base send time out to recover transaction by Se self. So this time is too much. Time out settings determined from server settings by default in mysql is 30 second.

C. ***Provider time out or server late in response for any causes after send response to coordinator.*** Coordinator knows web service time for each provider to get response. if one of them not response coordinator sends abort to another provider and tells client

D. ***Provider crashes within coordinator forward response to client or after client receives response from coordinator: -*** Here providers get final decision from coordinator so provider release resources but client can`t gets result.

Crashing coordinator

In proposed model coordinator embedded in UDDI server, therefore new supposed UDDI can`t crash. If coordinator server crashed, then request can`t transfer. In traditional model BPEL embedded in client server or separated server refers to client. Backward for proposed model client wait to get response from coordinator but if coordinator crashed, then client don`t know. Coordinator may be crash after forward request so provider here wait to get final decision but it can`t so provider take all default time to release row.

IV. CONCLUSION

A proposed model decreases locking time for provider's recourses in composite web services transaction model based on atomic transaction. Atomic means all providers must response else abort all transaction. Proposed model has three types of participant (client, coordinator, providers). First part is client. Client is a system has composite web service and tries to get atomic response from all web services participant, second partition is coordinator. Coordinator is middle system between requester (client) and responder (provider). Proposed model depends on coordinator is built in UDDI. Coordinator has error detection system. This error detection system handles transaction between providers and between providers with client. Coordinator supposes client need atomic transaction result. Third part is providers. Each provider is a system sharing web service. The drawback of proposed model is that if UDDI crashed no technique for detecting that. The experiments tests crashes of three parts in proposed model (client, coordinator, providers). The experiments tests crashes after and before sending request per part and also receives request to coordinator and client. Experimental results reveal that proposed SOA model has ability to limit locking time in case crashes of providers and client for atomic transaction model.

REFERENCES

[1]. Y.Ling,G.Lin.(2008).Research of Business Transaction Process in SOA Environment. International Conference on Computer Science and Software Engineering (p.p 1256 – 1259).

[2]. E.Joyce,M.Maude,R.Guillermo,R.Marta.(2008).QoS-driven Selection ofWeb Services for Transactional Composition.IEEE International Conference on Web Services (pp 653 – 660).

[3]. Cabrera,F.,Copeland,G.,Cox,B.,Freund,T.,Klein,J.,Storey,T.,and Thatte,S.(2002). Web Services Transaction (WS-Transaction) http://www-106.ibm.com/developerworks/library/wstranspec/

[4]. S. Tai,R. Khalaf,T.A. Mikalsen.(2004).Composition of Coordinated Web Services. 5th ACM/IFIP/USENIX international conference on Middleware(Vol. 78,pp 294 – 310).

[5]. L. Mikel,F. Antonio,andez,A. Sergio.(2002).Optimal Implementation of the Weakest Failure Detector for Solving Consensus. 19th IEEE Symposium on Reliable Distributed Systems(pp. 52 – 59).

[6]. Y. Weihai,W. Yan,Pu,C.(2004).A Dynamic Two-Phase Commit Protocol for Self-Adapting Services.IEEE International Conference on Services Computing(pp 7 – 15).

[7]. B. Boualem,Toumani,F.(2004).Analysis and management of Web Service protocols.Proceedings of the 23rd International Conference on Conceptual Modeling(pp524 – 541).

[8]. Y. Muhammad,M. Kuo,C. Chi,L. Yinsheng.(2006).An Efficient Transaction Commit Protocol for Composite Web Services.20th International Conference on Advanced Information Networking and Applications (Vol. 1,pp 591 – 596).

[9]. Z. Wenbing.(2006).Failure Resilient Distributed Commit for Web Services Atomic Transactions.This work was supported by a Faculty Startup Award and a Faculty,Research Development Award at Cleveland State UniversityPublication. eprint arXiv:cs/0612083 12/2006.

[10]. S. Changai,A. Marco.(2007).Requirements and Evaluation of Protocols and Tools for Transaction Management in Service Centric Systems.31st Annual International Computer Software and Applications Conference(Vol. 2,pp 461 – 466).

[11]. A. Mohammad,B,Wolf-Tilo,D. Peter Dolog,Nejd.(2007).Nonblocking Scheduling for Web Service Transactions.Fifth European Conference on Web Services IEEE(pp 213 – 222).

[12]. J. Pawel,X. Li,.(2008).Adapting Commit Protocols for Large Scale and Dynamic Distributed Applications.OTM Confederated International conferences,CoopIS,DOA,GADA,IS,and ODBASE Part I on On the Move to Meaningful Internet Systems.(Vol. 5331,pp 465).

[13]. Z. Wenbing,E. Louise,Moser,P. M. Melliar-Smith.(2008). A Reservation-Based Extended Transaction Protocol. IEEE Transactions on parallel and distributed systems (VOL. 19,NO. 2,pp 188 – 203).

[14]. R. Hossein,A. Hassan.(2008).Composite Web Service Failure Recovery Considering User Non-Functional Preferences.4th International Conference on Next Generation Web Services Practices IEEE(pp 39 – 45).

[15]. E.Korany, F.Gadelrab.(2010).proposed SOA Model for Unlocking Transaction Resources. ICCTA 2010,23-25 October 2010,Alexandria,Egypt