New SOA Model for Unlocking Transaction Resources

[1]Ezzat A.Korany, [2]Fatma Sayed Gad EL Rab.

[1] Prof of computer science.

[2] Student in Institute of Gradate Study and Research, Department of information technology, Alexandria University

Internet Technology Research, Alexandria, Al Shatby, 163 Elhoria road, Zip code: 21526, Egypt,

E-mail:f_s_g2001@yahoo.co.uk

***Abstract* — Successful e-business implementation begins with the creation of an appropriate structure for running an e-business project. This structure must be designed to ensure successful delivery for the eventual application of the e-business initiative, and must be capable of delivering the desired business functionality in a timely manner and avoiding the high failure rates. Dynamic e-business systems contain an E-business process that is usually developed by composite web service. Unexpected behavior from a component Web service may not only lead to its failure, but also may bring negative impact on all the participants of the composition. There are some protocols that have recently been developed for (composite) web services transactions like WS-Transactions, OASIS Business Transaction Protocol (BTP).Mainly these protocols use ACID properties and 2PC to handle transactions these lead to problems in performance and lock resources for long time .This paper surveys these different protocols and introduces a new solution to cover these problems but it is not implemented yet.**

***Index Terms* — web service, composite web service, and E-business transaction system, E-business protocols, SOA model.**

I. INTRODUCTION

E-business technologies are composed of one or more solution architectures. To start with Internet publishing then appears integration between systems (internal and external system) until dynamic e-business. Dynamic e-business can also utilize the developing web services standards to automatically locate and outsource business processes in real time. Application development using web services emphasizing the creation of software as an interconnected set of software components [1]-[2].A Web service is a specific kind of service that is identified by a URI but they require special consideration as a result of using a public like low fidelity mechanism for inter-service interactions. Services come in two flavors: simple (stateful services) and composite services (stateless services). Composite services involve assembling existing services that access and combine information and functions from –possibly- multiple service providers [3].Transactions exists to ensure that all parts of a particular business operation are properly recorded. If any single part fails, it should lead the transaction as a whole to fail in order to maintain data consistency [4]. But in disadvantage of web service It can`t manage transactions to handle this problem in composite web services using the advanced transaction model not conventional transaction model to ensure data consistency. Because of its long running and loosely coupled nature [5], some protocols have recently been developed for (composite) web services transactions. Like 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 are 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 in 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 also allow component transactions of a root transaction, to commit unilaterally irrespective of the commitment of their sib-ling transactions [6] - [7].

The purpose of this paper is to introduce a solution that achieves ACID properties in composite web service for each transaction taking into account the performance

problem that happened as a result of using 2PC in case of the crash of at least one of the web service providers but this new solution is not implemented yet. The rest of this paper is organized as follow. In section 2 some recent related work is briefly reviewed, Section 3: concept of new solution (Methodology), Section 4: Conclusion and future work, In section 5: acknowledgement, at last in section 6: References.

II .RELATED OF WORK

For centric system (SC) S.Changai et, al .grouping the requirements with respect to ACID properties and adding a fifth set of properties which goes beyond ACIDity. 1.0 Atomicity, 1.1 Rollback, 1.2 Compensating, 2.0 Consistency, 2.1 Abort, 2.2 Adding deadlines to transactions, 2.3 Logical expressions for specifying constraints, 3.0 Isolation, 4.0 Durability, 5.1 Composite transactions, 5.2 Distributed transactions, 5.3 Transaction recovery by dynamic rebinding and dynamic re-composition at runtime, 5.4 Secure transactions of different types (Confidentiality, Integrity, Authentication and Nonrepudiation), 5.5 Optimistic or pessimistic concurrency control. Denoting the satisfaction with the 'y', symbol, the partial satisfaction with 'p', and no support with 'n' as illustrated in table1.

BTP is not part of the WS-Stack, which limits its compatibility with other Web service technologies. In addition, BTP does not support long-lived transactions. There is also a difference in granularity between the above transaction standards.WS-AT contains simple two phase commit protocols, WS-BA contains non-blocking protocols and BTP consists of a sequence of small atomic transactions. Dynamic rebinding is supported only by BPEL, though only at the implementation level.

WSCDL supports most requirements, while its major disadvantage is that the large players in the field do not support it and that no implementation is available. WS-AT is a very conservative business transaction model especially with respect to blocking. WS-BA is more appropriate for services, by renouncing to the concept of the two-phase commit. BTP places itself in the middle (two phase commit is followed in a relaxed way). As for BPEL and WS-CDL they address the business process perspective with limited transaction support.

Reqs	BTP	WS-AT	WS-BA	BPEL	WS-CDL
1.0	y	y	n	p	p
1.1	y	y	p	p	y
1.2	n	n	y	y	p
2.0	y	y	p	p	p
2.1	y	y	y	y	n
2.2	n	y	y	p	y
2.3	n	n	n	y	y
3.0	n	y	y	y	y
4.0	y	y	y	y	p
5.1	y	y	y	y	y
5.2	y	y	y	y	y
5.3	n	n	n	y	n
5.4	n	n	n	p	p
5.5	n	n	n	n	y

Table 1 Evaluation Results

[8].Z.Wenbing et,al .Implement new reservation protocol that described a novel reservation based extended transaction protocol that can be used to coordinate the tasks of long-running business activities each task is executed as two subtasks. The first subtask involves an exclusive blocking reservation of the resource. The second subtask involves the confirmation or cancellation of the reservation. The reservation at the end of the first subtask becomes visible to other business activities, because fewer resources are available for them to reserve. However, this visibility does not Compromise the isolation property, because the reservation can be confirmed or cancelled and the other business activities cannot make any assumptions about resources that have not been reserved for them. For the duration of the reservation, the supplier grants an exclusive right to the client for the amount of goods reserved. During the second subtask, the reservation is confirmed only if the business activity can be completed successfully. Reservation protocol involves two phases.

There are a number of differences between this protocol and 2PC, in this protocol, the reservation of a resource is executed as a traditional ACID 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, which has no control over how long the resource is locked Another difference between this reservation protocol and locking is the effect on other transactions that need to access the resource. If a resource is reserved and another transaction wants to access it, the transaction can acquire a lock on the resource, and the application can be informed immediately of the state of the resource (that is, some of the resource has been reserved, but a sufficient quantity of the resource remains to satisfy the reservation). Thus, the application can take an appropriate action without delay. However, if the resource is locked by the database system and another transaction wants to access it, the new

transaction must wait until the lock is released the waiting time might be long, in which case the application cannot take immediate action. Once again, this characteristic is not unique to the reservation protocol. The escrow transactional method also has this characteristic [9].

I see this solution is not suitable in case of failure of one task, like if it fails in the payment task after it releases recourses in the reservation task to grant another client access to the same record. So it has implemented a compensation process to rollback the previous reservation, but it rolls back after another client does a transaction. So I see this solution do treat locking problem.

Yu.Weihai et, al construct an improved 2PC (Called Dpr Dynamic presumption) it is like 2PC but it consists of a voting phase and a decision phase. This new feature

• A coordinator maintains in main memory a protocol table (PTbl) that contains an entry for every transaction that has entered 2PC. For every transaction, it maintains a list of participants, their presumptions and votes (if voted).

• The messages WorkDone, Yes and log record Prepare contain the presumption-bit of the participant. The messages prepare (if the coordinator can override the presumptions), Commit, Abort and the coordinator log record Commit contain presumptions of all participants.

• Before the protocol starts, the coordinator has received WorkDone messages from all participants and therefore is aware of their preferred presumptions. The coordinator may choose to override the preferred presumptions of the participants and include this information in the Prepare message. .

• The coordinator maintains a forced-availability window for all PrC participants (if any) and a forced-availability window for all PrA participants (if any). Care is taken to properly close all forced-availability windows so that the log can be correctly garbage collected.

• Each participant includes the presumption- bit in the Prepare log record which opens an in-doubt window. Care is taken to properly close all in-doubt windows, so that the log can be correctly garbage collected.

• The participants only acknowledge the outcomes that are different from the presumed ones. The duplicated outcome messages are also acknowledged. (The messages include the presumptions of participants, so the participants know the presumptions even when the in-doubt windows are closed.)

• On recovery from a system crash or on timeout of an expected outcome message, a participant inquires the outcome by re-sending the Yes vote message. The message includes the presumption-bit.

• The coordinator responds to the participant with the presumption included in the Yes vote message when no information about the transaction is available [10].

Younas.M et, al. presents 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. TCP4CWS is based on the assumption that the desired services have already been discovered using existing techniques such as UDDI.

Case	T_{msg} (ms)	T_{fw} (ms)	$P(cst_i)$ (ms)	2PC	PA		TCP4CWS-1		TCP4CWS-2	
				Commit/Abort Exp (2)	Commit Exp (2)	Abort Exp (3)	Commit Exp (4)	Abort Exp (5)	Commit Exp (6)	Abort Exp (7)
1	100	20	100	580	580	420	340	440	560	660
2	300	20	200	1420	1420	1120	840	1040	1360	1560
3	100	20	300	760	760	620	540	840	960	1260
4	300	40	300	1620	1620	1240	980	1280	1620	1920
5	300	40	400	1720	1720	1340	1080	1480	1820	2220

Table 2 Time required by 2PC, PA, and TCP4CWS to process TCW

P (csti) is the processing time of a component web service transaction (csti). Tmsg is the time taken to communicate message between composite web service coordinator (CWC) and sub coordinator (SC). Tfw is the time taken in forced write operation. Forced-write operations affect performance and are therefore taken into account. In these operations, CWC and CS, 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 (global decision from CWC but if commit form SC and finally receive abort message from CWC it cancel process. Table 2 uses the above values of P (csti), Tmsg, and Tfw to calculate the commit and abort delays of the protocols in five different cases. Using the above expressions (1) – (7), these delays are calculated with the combination of minimum and maximum values of P (csti), Tmsg, and Tfw.TCP4CWS-1 and TCP4CWS-2 respectively represent the proposed protocol with and without alternative csti.

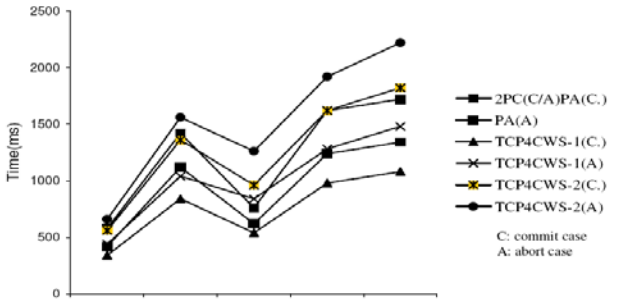

Fig 1 Comparison of the protocols' delays required to commit/abort TCW

Figure 1 graphically represents the commit/abort delays incurred by the protocols under consideration. The proposed protocol

TCP4CWS-1 outperforms 2PC and PA in the case of TCW's commit. There is a noticeable difference between TCP4CWS-1 and 2PC/PA when TCW is committed. TCP4CWS mainly optimizes the commit delay due to the unilateral commit strategy, which results in fewer messages communicated between SCs and CWC.In the case of TCP4CWS-2 with alternative csti, it still performs better than 2PC and PA provided the message delay is high (case 2). However, if the processing delay is higher, then TCP4CWS results in poor performance due to executing alternative transactions and cancellation of services through compensating transactions. Though the use of alternative transactions affects the performance, they significantly increase the commit rate of TCW. Due to space limitation they cannot show the results how alternative transactions increase the commit chances of TCW.

The use of alternative transactions is also essential in the composite web services as there are exists various alternative services. Further PA outperforms 2PC in the abort cases, as it reduces the number of messages and forced write operations. It also performs better than TCP4CWS in all abort cases except in case 2 where the message delays are higher. However in commit case it does not performs well as described above. For most applications it is necessary to improve the performance of transactions in the commit case as it is more desirable to commit a transaction than to abort a transaction. TCP4CWS is built on this notion to improve the performance in the commit case of transactions [11]. In case of abort any SC CWC send final abort to all SC so this resources of SC still locked until receive final message so there are performance issue

J. Pawel et, al present a distributed commit protocol for supporting a wide variety of applications. The protocol has a number of features distinguishing it-self from existing solutions. First, the protocol addresses both small scale systems with a handful of nodes and larger systems with hundreds of nodes. Second, it is resilient to network partitioning and multiple node failures. Finally, the protocol provides an exible solution in the level of consistency through adjustable parameters and o_ers a trade-o_ between consistency and e_ciency. New commit protocol based on 3PC that is scalable and resistant to dynamic network and node failures, and provides a con_gurable level of consistency depending on speci_c application and system deployment characteristics [12].

A.Maha et,al. present a non-blocking atomic commitment protocol, noted ANB-CLL (Asynchronous Non-Blocking Coordinator Logical Log), that drastically reduces the cost of distributed transaction commitment in terms of time delay and message complexity. Performance analysis shows that the resulting protocol is more efficient than all other non-blocking protocols proposed in the literature. An important characteristic of ANB-CLL is that it can be applied to commercial transactional systems that are not 2PC compliant. To achieve non-blocking, ANBCLL uses a uniform consensus protocol as a termination protocol in an asynchronous system augmented with an unreliable failure detector, and in which processes may crash and recover by supporting recovery.

Comparing ANB-CLL with the DNB-AC and MD3PC protocols (these are the most well-known non-blocking protocols that were proposed in the context of asynchronous systems). Furthermore, and for the sake of completeness, they also make a comparison with the standard 2PC and 3PC protocols, which are undoubtedly the reference point in the context of atomic commitment.

Protocol	Number of steps	Number of Messages *Without broadcast network*	Number of Messages *With broadcast network*
2PC	3	3n	n+2
3PC	5	5n	2n+3
DNB-AC	3	n (2n+1)	2n+1
MD3PC	3	3n (f+1)	n+f+2
ANB-CLL	2	n (n+1)	n+1

Table 3 Latency and Message Complexity for the different protocols

Table 3 shows the performances of the different protocols in the absence of failure suspicions, with and without a broadcast network. Performances are given in terms of latency (i.e. number of steps needed to commit) and message complexity (i.e. the number of messages needed till a decision is reached on the participants). (n) indicates the number of participants in the transaction (including the coordinator), while (f) represents the resiliency rate on which the MD3PC protocol is based. As pointed out in the introduction, 2PC, DNB-AC and MD3PC have the same latency (3 communication steps), while 3PC requires 5 communication steps. From Figure 2, it is clear that ANB-CLL is faster than all the above protocols as it only needs 2 communications steps. Concerning message complexity, they distinguish two cases: (1) with a broadcast network, and (2) without a broadcast network. In case (1), and assuming 3 participants in the transaction (n = 3) and a resiliency rate of 2 (f = 2), 2PC needs 9 messages, 3PC needs 15 messages, DNB-AC needs 21 messages, MD3PC needs 27 messages, and ANB-CLL needs 12 messages. In case (2), 2PC needs 5 messages, 3PC needs 9 messages, DNBAC needs 7 messages, MD3PC needs 7 messages, and ANB-CLL needs 4 messages. As a conclusion, ANB-CLL is significantly faster than all other protocols, and in case of a broadcast network, it even decreases communication overhead. This high efficiency makes ANB-CLL very well adapted to the needs of today's advanced systems [13].

P.Alberto et, al. propose some aspects for characterizing the transactional behavior (key dimension can be used for identifying transactional behavior in services based applications) and furthermore introduced a critical review of current approaches. For analyzing approaches providing transactional behavior. These are key dimensions (table 4).

Duration is related to transaction lifetime. for characterizing shared resources locking it divides to short (millisecond for finish)and long(hours or days or week to finish) at the most process take long duration , atomicity is based on the principle of "all-or-nothing" for characterizing execution and associating semantics to the transactions in presence of exceptions there are Three types of atomicity can be considered: Strict atomicity is(classical definition) a transaction is treated as an execution unit which can be completely executed or not, Semantic atomicity introduces the concept of compensation.

Semi atomicity enables transaction committing when a primary set of predefined operations commits or when an alternative set of predefined operations commits.

	Dimension	Values
1	Duration	- Short
		- Long
2	Atomicity	- Strict atomicity
		- Semantic atomicity
		- Semi atomicity
3	Isolation	- Local
		- Global
4	Control flow	- Implicit
		- Explicit

Table 4 Dimensions for characterizing transactional

Isolation is related to visibility degree of results within a transaction to other concurrent transactions. For characterizing in which degree resources can be shared among transactions Isolation can be either, local or global. Local isolation enables the knowledge of partial results within the transaction but it cannot reveal its results to other concurrent transactions before it commits. Therefore resources remain blocked during the transaction lifetime. Global isolation enables the knowledge of partial results between several concurrent transactions and the access to common resources before committing. Global isolation is needed when data is distributed and transactions are long duration. In process oriented approaches, local isolation is related to execution units while global isolation is related to the whole transaction. Control flow specifies the execution order of operations within transactions, it for characterizing the execution strategies of several concurrent transactions. Operations are related not only to queries, but to complex business processes that involve computations in several sites. It can be either implicit or explicit. Implicit control flow is hard coded within the transaction (i.e. sequential execution). It is normally encompassed within the execution logic of transaction. Explicit control flow is specified by the developer as a part of the transaction definition. For example, to commit semi atomic transaction it is necessary to specify an alternative set of operations along with a preference order. Explicit control can be defined imperatively or declaratively.

Regarding atomicity, providing just one type of atomicity is not enough when multiple participants and context are involved. Furthermore the application characteristics determine the type of atomicity required. Isolation is related to the degree of visibility of partial results. Local or global isolation must be provided depending on application needs. Control flow in current approaches is addressed either explicitly or explicitly. They believe that control flow must be provided implicitly in the model with the possibility of modifying. Implicit control flow can address well known atomic models while explicit control flow can address specific applications needs such as parallel recovery, selective compensation, etc.

The first step for providing transactional behavior to information systems was to provide transactional behavior to data centric applications. They adopt ACID properties for managing data by means of centralized and distributed transactions the aim of distributed transactions is to share data trying to minimize blocking time. This can be achieved by using well know protocols of commitment such as two-phase commit (2PC) and advanced transactional models. Advanced transactional models have been proposed as a way to tackle distribution issues. The key principle of such models is to divide transactions in short running sub transactions. A sub transaction can export its results as soon as it commits but if something goes wrong or changes, it is necessary to amend its effects by using compensating transactions.

A compensating transaction is a “semantic undo”. In particular they are interested in advanced transactional models because they introduces the notion of how executing operations within transactions as a part of the definition itself of transactions. Three well known proposals of them are saga, flexible transactions, and contracts. Transactional behavior for current applications is addressed by process oriented approaches (i.e. transactional workflows). In such approaches, transactions are used to ensure consistency among computations using process as execution units. They deeply analyze the following approaches that provide transactional behavior to business processes: compensation and atomicity spheres approach that introduces the concept of control spheres , OASIS-BTP business transactions protocol that address the problem of coordinating business processes with transactional properties , Web services transactions that addresses transactional behavior to Web services , and two models addressing atomicity for coordination of Web services based on tentative-hold protocol and patterns .

	Approach	Duration		Atomicity			Isolation		Control flow	
		Short	Long	Strict	Semantic	Semi	Local	Global	Implicit	Explicit
Data centric	Saga		◊		◊		◊		◊	
	Flexible transactions		◊			◊	◊			◊
	Contract		◊			◊	◊			◊
Process oriented	Spheres		◊			◊	◊		◊	
	BTP	◊	◊	◊		◊				◊
	WS-C+WS-Tx	◊	◊	◊	◊			◊	◊	◊
	Model based on THP		◊	◊		◊		◊	◊	
	Model based on patterns		◊	◊		◊		◊	◊	

Table 5 Approaches with respect to dimensions

Table 5 summarizes the presented approaches according to four dimensions that mentioned before. Note that most of the approaches do not address all values of dimensions. While data centric approaches are mainly concern of short duration transactions with local isolation, process oriented approaches are concern of long duration transactions with global isolation. Beside most of the approaches address only one kind of atomicity assuming either implicit or explicit control flow. After analyzing Table 4 they conclude that transactional behavior has been tackled using ad-hoc strategies. They think that is possible to address all values of dimensions in existing service based applications. Consequently, they propose an approach that separates the specification of application logic and the specification of transactional behavior as follows:

• Application logic must be captured by using a successful coordination approach (i.e. workflow technology).

• Transactional behavior must be defined by using atomicity contracts and associating a well defined behavior to participants of coordination [14].

R. Hossein et, al introduce a heuristic distance measure which significantly reduces search space of hybrid (i.e. forward-backward) search algorithm and results in near-optimal solutions for composite web service, services are more susceptible to failures This is due to its dependency on other services which are external modules to the composite service MAX_MIN heuristic distance measure for estimating the distance between two sets of literals (i.e.

service's input/output). Each literal represents one of the members of service's Input/output set.

The distance measure is used for reducing the search space of exhaustive search algorithm. Heuristic distance measure reduces search space significantly and results in so near-optimal solutions. They proposed a mechanism to gather some knowledge in well defined data structures in offline. Then it uses those extracted knowledge for heuristic calculation. Heuristic calculation is kind of general MAX_MIN algorithm and is used when distance between two sets of literals is needed. At runtime, they ask user non-functional properties (i.e. Cost, Time and Reliability) precedent, then augment the MAX_MIN algorithm with user's preferences. Experiment results show that, although proposed method is not optimal, but it is near-optimal with significant reduced search space [15].

Z. Wenbing report mechanisms for A distributed transaction that might not commit atomically at correct participants if there are more faults and implementations in the context of a Web services atomic transaction framework that significantly increase the probability of atomic commitment of distributed transactions even when the majority of coordinator replicas become faulty. The main novelty of our design is the minimized runtime overhead and the increased failure resiliency of distributed commit under Byzantine faults. The core mechanisms include a piggybacking mechanism, which limits the way a faulty coordinator replica can do to cause confusion among correct participants, and a voting mechanism, which enables fast agreement on the transaction outcome under fault-free situation, and ensures that the agreement is based on the messages from correct replicas with high probability even if all but one coordinator replica becomes faulty. Their performance study on an implemented prototype system shows only 10% end-to-end runtime overhead under both fault-free and faulty scenarios. This proves the practicality of their mechanisms for use in real-world Web-based transactional systems.

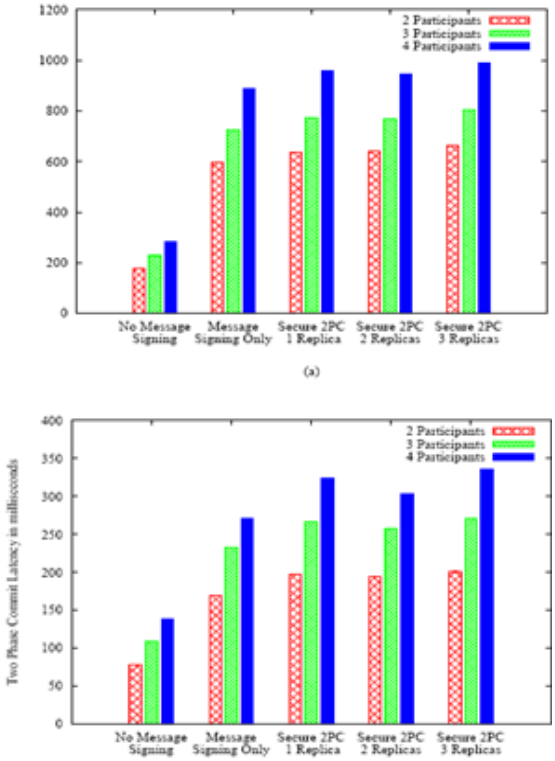

Fig 2 The measurements of the end-to-end latency (a) and the two-phase commit latency (b) under different fault –free scenarios.

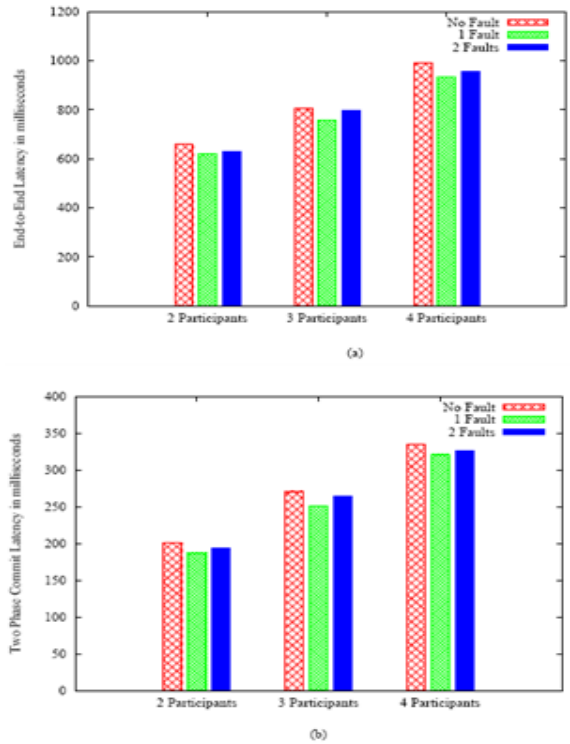

Fig 3 the measurements of the end-to-end latency (a) and the two-phase commit latency (b) under different number.

[16]. A.Mohammad et, al .proposed a novel no blocking scheduling mechanism that is used prior to the actual service invocations. Its aim is to reach an agreement between the client and all participating providers on what transaction processing times have to be expected, accepted, and guaranteed. This enables service consumers to find a set of best suited providers fitting their deadlines. Service providers on the other hand can benefit from the proposed mechanism due to the now possible intelligent scheduling of service invocations for best throughput.

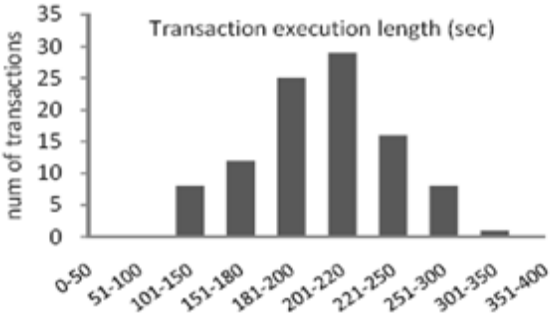

Fig 4 Distributed of transaction`s length

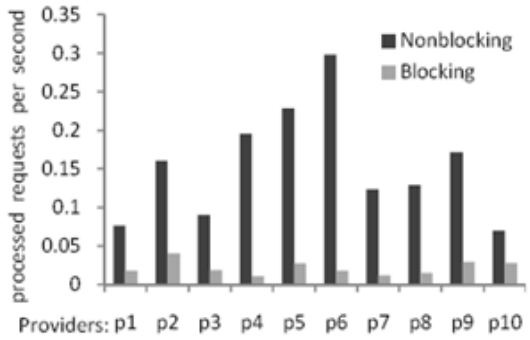

Fig 5 Resources utilization

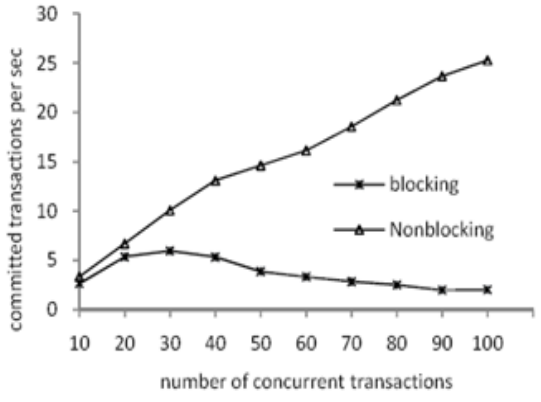

Fig 6 Overall throughputs

The proposed solution enhances conventional scheduling algorithms for concurrency control to support the time characteristics of the transactions in Web service environment. By applying the proposed scheduling mechanism unnecessary blocking of transaction commitment during the execution of a 2PC protocol is avoided, thus saving time and costs of later abort or missed deadlines. The proposed approach is beneficial for both the service consumer and service provider. The experimental results showed a significant improvement in terms of number of successfully completed transactions within acceptable time frames as well as in terms of resources utilization [17].

L. Mikel et, al. present a new algorithm implementing 3S. there algorithm guarantees that eventually all the correct processes agree on a common correct process. This property trivially allows us to provide the accuracy and

Completeness`s properties required by 3S. They show that there algorithm is better than any other proposed implementation of 3S in terms of the number of messages and the total amount of information periodically sent. In articular, previous algorithms require to periodically exchange at least a quadratic amount of information, while ours only requires O (n log n) (where n is the number of processes).However, they also propose a new measure to evaluate the efficiency of this kind of algorithms, the eventual monitoring degree, which does not rely on a periodic behavior and expresses better the degree of processing required by the algorithms. They show that the runs of their algorithm have optimal eventual monitoring degree [18].

III. METHODOLOGY

This methodology presents the design of a new architecture model for web service systems and how a new model handles transactions for consumers that use composite web service to decrease lock time of provider resources in case of crash. Previous model needs client build coordinator to handle transactions between web services in composite web service or use any of standards like BTP as mentioned before. There are backwards for these standard like using 2PC to handle atomic transaction, in case of a consumer crash or coordinator crash, it is in a separated server. These lead to the lock of provider resources. So this new model doesn`t need neither an external coordinator nor 2PC. At the same time, composite web service is atomic transaction.

Dear reader you will see

• General view of new model of WS (System design).

• Detailed View of relations between NewUDDI (NUDDI) and Consumer.

• Detailed View of relations between NUDDI and Providers.

• Detailed View about NUDDI How NUDDI handles transactions In case of 3 cases.

- Normal case (No failure)
- Failure of Consumer
- Failure of provider

3.1. General view of new model of WS (System design):-

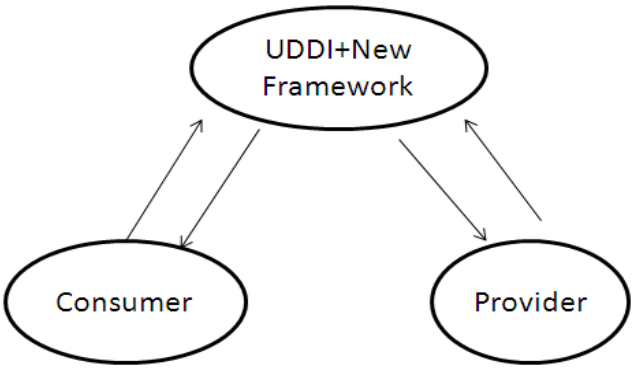

Fig 7 New model of web service architecture

Figure 7 illustrates new model of web service architecture. Provider can publish service as before but there are differences in contact consumer with provider.

3.2. Detail View of relation between NewUDDI (NUDDI) and Consumer.

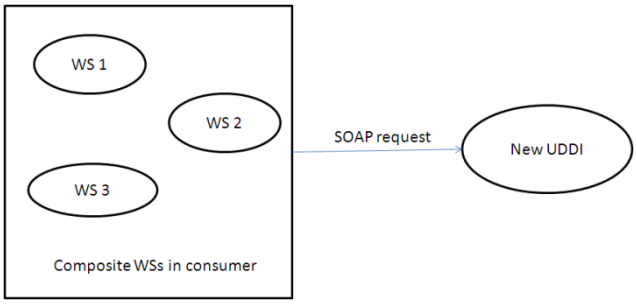

Fig 8 Consumer send SOAP request to NUDDI

• Consumer query WSDL and check if it available or not.

• Consumer send SOAP request to UDDI, this request includes Composite WS (consumer want some of WSs).

3.3. Detail View of relation between NUDDI and Providers.

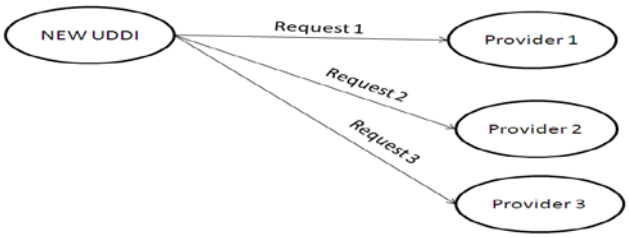

Fig 9 NUDDI forward request to provider

• UDDI send request per provider (Figure 9).

•Each provider response to NewUDDI after it completes

•Implementation (Normal case without the crash of any provider).

• NUDDI forwards these responses to consumer (normal case without crash consumer).

3.4. Detail View about NUDDI How NUDDI handle transaction In case of 3 cases.

As mentioned before WS new model was designed to treat backwards in 2PC in case of failure that lead to lock resources of providers .previous model has 3 failure scenarios:-

• **Failure of consumer**:-this scenario illustrates how consumer can`t tells coordinator to commit.

So if this crash was discovered early (immediately crash) provider would not need to lock its resources. These times starting from receiving a request to receiving a commit or abort from coordinator. So, in the new model, there is a session opened between coordinator and consumer, if it expires for any reason, the coordinator immediately sends abort to all providers to cancel the process and release its resources.

• **Failure of coordinator**:-some servers that have coordinators may crash for any reason like network...etc.

So, providers lock resources utile receiving a commit or abort for data consistency. So in the new model, the coordinator is built in the UDDI server.

• **Failure of provider**: - As we know, A composite webservice is a service. Some web services, after sending a request, sometimes one of these providers' crashes. So, the atomicity and consistency transaction coordinator has to send abort to all other providers. But this abort message will be sent after all providers finish execution. So, this is a huge time. Specifically that provider that crashed has greater time to finish its execution than others. The new model tries to treat this problem by doing 3 things: *First* get time per WS function from WSDL, *second* NUDDI use DISO frame work [20] to calculate latency in network between provider and NUDDI server so each function has 2 times WSft (web service function time) +NLt (network latency time) So provider has to respond after WSt=WSft+NLt. *Third* NUDDI calculates Max time for all WSs like WSt1<WSt2<WSt3 so here max is WS3 provider has to be publish time per web service function in WSDL. This time will be used by the new coordinator built in NUDDI to calculate time per exaction

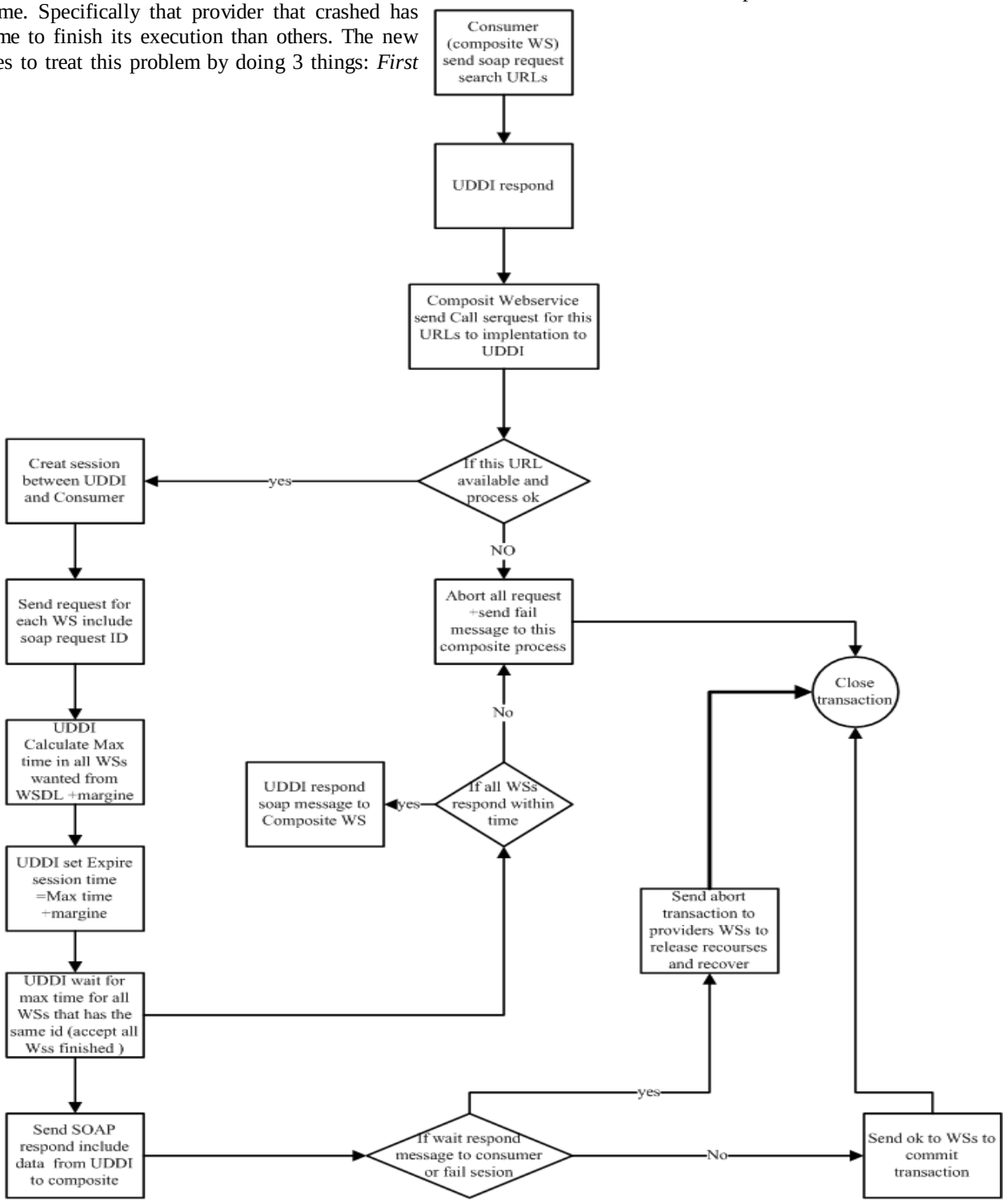

Fig 10 Detail NUDDI

IV .CONCLUSION AND FUTURE WORK

Ensuring consistency and atomicity of composite web service in any E-business system is essential. It is not enough to grantee good performance of this system. Time is an important item that has to be taken into account in e-business. The new model of WS system try to take time into account and handles transactions in a new approach. In this paper UDDI becomes the handler of the transaction to grantee the reduction of the likelihood of coordinator crash, NUDDI opens session between UDDI server and consumer to make sensor to consumer. If it crashes at any time during execution of WS, UDDI aborts all transactions to release recourses of the provider. Also calculates time that each WS would be implemented + time to transport data over the network. If any WS Overtakes this time, UDDI starts calculating time that it will consume to re call and get new response. If it is greater than max time of WS in this composite, WS UDDI sends abort to all other transactions to release resources. In future work the new solution will be implemented and compared with old model of SOA using BTP and web service transaction in locked time of resources in case of failures of at least one of the provider or consumer .

ACKNOWLEDGEMENT

The authors wish to acknowledge the assistance and support of the ICCTA Steering Committee.

REFERENCES

[1] K.Ravi, R.Marcia,"E-business 2.0 Roadmap for Success" *Copyright 2001 addison-Weseley*

[2] W.Dougal,"E-business Implementation A guide to web services, EAI, BPI, e-commerce, content management, portals, and supporting technologies" ,*First published 2002 Butterworth-Heinemann* ,An imprint of Elsevier Science Linacre House, Jordan Hill, Oxford OX2 8DP

[3] C.Lawrence, M.Zakaria, M.David, B.Boualem, "Extending Web Services Technologies The Use of Multi-Agent Approaches" *Library of Congress Cataloging-in-Publication Data 2004 Springer Science Business Media, Inc.*

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

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

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

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

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

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

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

[11] Y. Muhammad,M. Kuo,C.Chi,L. Yinsheng,"An Efficient Transaction Commit Protocol for Composite Web Services", *20th International Conference on Advanced Information Networking and Applications (AINA'06) ,Vol. 1,pp* 591 – 596, April 2006.

[12] J.Pawel, X.Li,"Adapting Commit Protocols for Large-Scale and Dynamic Distributed Applications" OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part I on On the Move to Meaningful Internet Systems,Vol. 5331,pp 465 - 474 ,2008.

[13] A.Maha, P.Philippe,"A Low-Cost Non-Blocking Atomic Commitment Protocol for Asynchronous Systems" *12th International Conference on Parallel and Distributed Computing and Systems (PDCS)*, USA, November1999.

[14] P. Alberto,V.Genoveva ,Z. Jose-Luis ,C. Christine, G.Luciano "A survey for analyzing transactional behavior in service based applications" *Seventh Mexican International Conference on Computer Science (ENC'06) ,pp* 116 – 126,2006

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

[16] Z.Wenbing, "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 University*Publication**.**eprint arXiv:cs/0612083 12/2006

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

[18] L.Mikel,F.Antonio,andez,A.Sergio,"Optimal Implementation of the Weakest Failure Detector for Solving Consensus " , *19th IEEE Symposium on Reliable Distributed Systems, 2000. SRDS-2000.* pp.52 – 59, References Cited**:** 11, August 2002.

[19] http://disco.informatik.uni-kl.de/