A Framework for Handling Dependencies among Web Services Transactions

Seunglak Choi
KAIST
371-1 Guseong-Dong, Yuseong-Gu, Daejeon, Korea
slchoi@dbserver.kaist.ac.kr

Hyukjae Jang
KAIST
371-1 Guseong-Dong, Yuseong-Gu, Daejeon, Korea
hjjang@nclab.kaist.ac.kr

Hangkyu Kim
KAIST
371-1 Guseong-Dong, Yuseong-Gu, Daejeon, Korea
hkkim@dbserver.kaist.ac.kr

Jungsook Kim
ETRI
161 Gajeong-dong, Yuseong-gu, Daejeon, Korea
jungsook96@etri.re.kr

Su Myeon Kim, Junehwa Song
KAIST
371-1 Guseong-Dong, Yuseong-Gu, Daejeon, Korea
{smkim,junesong}@nclab.kaist.ac.kr

Yunjoon Lee
KAIST
371-1 Guseong-Dong, Yuseong-Gu, Daejeon, Korea
yjlee@cs.kaist.ac.kr

ABSTRACT
This paper proposes an effective Web services (WS) transaction management framework to automatically manage inconsistencies occurred by relaxing isolation of WS transactions.

Categories and Subject Descriptors
H.2.4 [**Database Management**]: Systems – *Transaction processing*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *Distributed systems*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based services*

General Terms
Management, Design, Reliability

Keywords
Web services, Transaction model, Transaction management protocol, Isolation relaxation.

1. INTRODUCTION
Major IT organizations such as Amazon, Google, and e-Bay have been migrating their business interfaces to service-oriented architectures using Web Services (WS). For efficient processing, WS transactions commonly relax isolation property rather than preserve it by using traditional locking mechanisms such as the two-phase commit (2PC). Web services transactions can release locks on resources before their completions, and thus other transactions may access those resources [1][2][3][5].

However, the relaxed isolation introduces an inconsistency problem which can hinder the popular use of WS transactions. Suppose that a sub-transaction ST_a of a WS transaction T_1 completes while T_1 is not completed. A new transaction T_2 can use those data since the locks on the data updated by the ST_a are released. Now assume that T_1 is failed. Then, the update of the data should be cancelled, which is usually called *compensation*. This situation causes an inconsistency problem; T_2 should be aborted since it is dependent upon the updated value of the data. This problem can be serious in many situations such as financial and business environments. However, existing WS transaction models and managing systems do not address this inconsistency problem.

In this paper, we first formulate the dependency between transactions – named *completion dependency* – which could incur the aforementioned inconsistency problem. Then, we propose a new *Web services Transaction Dependency management Protocol* (WTDP) as an extension to de-facto WS transaction standards. Using WTDP, any changes affecting the status of the dependency are continuously monitored and processed by a WS transaction management system. Each organization is notified when all the relevant dependencies are resolved. Finally, we prototype a WS management system which supports WTDP.

Copyright is held by the author/owner(s).
WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

2. Completion Dependency
A *global transaction* (corresponding to a WS transaction) is composed of sub-transactions, where each *sub-transaction* is a Web services locating at a site. Actual operations such as database modification take place in the sub-transactions.

Definition 1 (Completion dependency): Let us assume that there are two global transactions GT_a and GT_b. The completion dependency $GT_b \Rightarrow GT_a$ occurs if sub-transaction ST_j reads a value updated by sub-transaction ST_i where $ST_i \in GT_a$ and $ST_j \in GT_b$.

In the above definition, we refer to GT_a as a *dominant global transaction* and GT_b as a *dependent global transaction*. We also refer to ST_i and ST_j as a *dominant* and a *dependent sub-transaction,* respectively. A dominant global transaction determines whether its dependent global transactions should be successfully committed or not. For instance, GT_b should be aborted if GT_a fails, because the update on the data referenced by GT_b has been cancelled.

A set of global transactions can be circularly connected by completion dependencies. We refer to a set of global transactions

having circular connection of completion dependencies as a *universal transaction*. This circular connection of completion dependencies may cause a serious problem. For instance, there could be a dead lock if global transactions of a universal transaction decided to delay their commitments until all the related global transactions are completed.

3. Web Services Transaction Dependency Management Protocol

Web services Transaction Dependency management Protocol (WTDP) is designed as an extension of the de-facto WS transaction standards - WS-Transaction and WS-Coordination protocols [3][4]. Technically, we add three services to coordinators as shown in Figure 1.

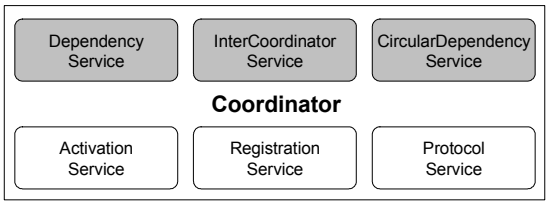

Figure 1 Services at a coordinator: gray-colored ones are the services provided by WTDP.

Dependency service is used for participants of a WS transaction to inform their coordinators of dependencies when they detect dependencies. *InterCoordinator service* is used to exchange dependency information among coordinators. For instance, the completion status of global transactions such as abort or commitment is delivered among coordinators using InterCoordinator service. *CircularDependency service* is used to determine whether there is a circular connection of completion dependencies. Note that, CircularDependency service is also used to resolve circular waiting in the universal transaction if all the transactions are delaying their commitments for the completions of other transactions. These services are defined as WSDL. Each service uses one or more messages to communicate with the services of other coordinators.

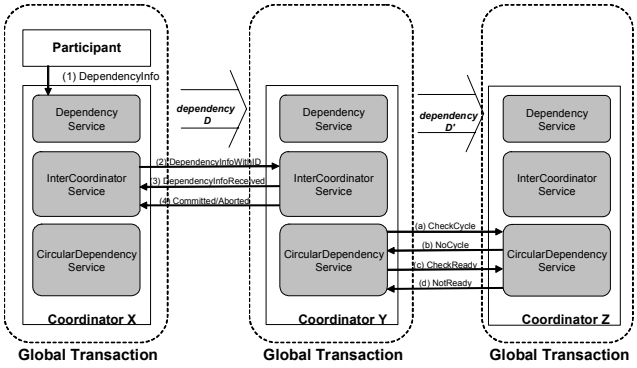

Figure 2. The message flow of WTDP. There are two completion dependencies – *D* and *D'*. *D* is between global transactions managed by coordinators *X* and *Y*. *D'* is between global transactions of coordinators *Y* and *Z*.

Figure 2 shows WTDP messages and their flows. When a completion dependency *D* is detected at a participant site, it informs a dependent coordinator of *D* by sending a DependencyInfo message (1). The dependent coordinator informs its dominant coordinator by using a DependencyInfoWithID message (2). The dominant coordinator acknowledges the message by returning a DependencyInfoReceived message (3). Now, the dominant coordinator sends Committed or Aborted messages to the dependent coordinator according to the completion status of the dominant transaction (4). The dependent coordinator can handle its global transaction appropriately according to the completion status of the dominant global transaction.

Meanwhile, the dominant global transaction may additionally have a completion dependency *D'* with another global transaction. In this case, the dependency relations may become a cycle. To determine whether a universal transaction is in the state of circular waiting, the dominant coordinator of *D* sends a CheckCycle message to the dominant coordinator of *D'* (a). The dominant coordinator of *D'* returns NoCycle if no cycle is found (b). Note that if a cycle exists, the CheckCycle message will finally be returned to the sender. Detailed description of all the messages and their flows are discussed in [6].

4. Prototype Implementation

To validate the proposed protocol, we implemented a prototype system which supports WTDP. It consists of a coordinator module and a participant module. All the WTDP services except a CircularDependency service are implemented. The prototype also supports most functions specified in the WS-Coordination/Transaction. We develop the prototype on the Linux operating system using Java. Both modules use Apache Tomcat 4.1.18 and Apache Axis 1.1 beta as a Web server and a SOAP engine, respectively.

5. Conclusions

In this paper, we formulated a completion dependency and proposed a Web services Transaction Dependency management Protocol (WTDP). For the easy deployment of WTDP in existing WS transaction management environments, WTDP is designed to be compatible with WS-Coordination/Transaction. We prototyped a WS transaction management system supporting WTDP. Currently, we are planning to address the completion detection issue at a participant site.

6. REFERENCES

[1] H. Garcia-Molina and K. Salem. SAGAS. In Proceedings of ACM SIGMOD Conference, pages 249-259, 1987.

[2] G. Weikum and H. J. Schek. Concepts and Applications of Multilevel Transactions and Open Nested Transactions. In A. Elmagarmid (ed.): Database Transaction Models for Advanced Applications, Morgan Kaufmann Publishers, 1992.

[3] IBM, Microsoft, and BEA. Web Services Coordination. http://www-106.ibm.com/developerworks/library/ws-coor

[4] IBM, Microsoft, and BEA. Web Services Transaction. http://www-106.ibm.com/developerworks/webservices /library/ws-transpec.

[5] OASIS. Business Transaction Protocol. http://www.oasis-open.org/committees/documents.php?wg_abbrev=business-transaction

[6] S. M. Kim, S. Choi, H. Jang, H. Kim, J. Kim, and J. Song. A Framework for Handling Dependencies among Web Services Transactions. Technical Report CS-TR-2004-207, KAIST.