Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
August 2009, 16(4): 59–66
www.sciencedirect.com/science/journal/10058885
The Journal of China Universities of Posts and Telecommunications
www.buptjournal.cn/xben

# Ensuring consistency of Web services transaction

WANG Xiao-jun[1](✉), LI Yang-qun[1], WU Xiao-mei[2], MIN Li-juan[1]

1. Institute of Information and Network Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
2. Foreign Languages Office, Nanjing Institute of Politics, Nanjing 210003, China

**Abstract**

Composite Web services provide promising prospects for conducting cross-organizational Web service transactions. Such transactions generally require longer processing time and manipulate financially critical data. To efficiently manage these Web services transactions, isolation is commonly relaxed, but inconsistency will be caused by concurrently executing isolation-relaxing transactions. This article proposes an extension to the WS-Business Activity Protocol, which ensures the consistent executions of isolation-relaxing Web service transactions, and which is based on transaction dependency graphs distributed over multiple nodes. Furthermore, this article presents several algorithms implementing the protocol, and introduces the implement of a prototype system.

**Keywords** Web services, transaction model, transaction management protocol, consistency

## 1 Introduction

Many IT organizations have been implementing interfaces for business partners using the Web services (WS) technology. Due to its flexibility, WS allows organizations to composite services across other organizations as well as within organizations. Composite Web services provide promising prospects for conducting cross-organizational Web service transactions. Such transactions generally require longer processing time and manipulate financially critical data. However, traditional protocols like the two-phase locking protocol and the two-phase commit protocol, are inappropriate for long running transactions. This is due to the fact that these protocols are based on the strict isolation policy which requires that a transaction should not share its data with other transactions until it is completed. Thus, if traditional protocols are used, a transaction may not be able to access its resources even for a few days, until other transactions release their resources. To efficiently manage these Web services transactions, isolation is commonly relaxed. But the isolation relaxation introduces a serious inconsistency problem. Considering a situation where several transactions ($T_1$ and $T_2$) concurrently invoke WS from different providers, one participant $P_1$ in $T_1$ needs to be compensated after it has updated its data item $X$, but another participant $P_2$ in $T_2$ has already read $X$ before $P_1$ is compensated and $X$ is changed back to its original value. In that case, the participant may be in conflict with another one from another transaction. Such a situation may occur quite frequently because WS transactions will get more and more prevalent [1]. Although most transaction models and protocols for WS transactions have been proposed, no sophisticated concurrency control mechanism exists. Thus, organizations must take the responsibility and make great efforts to solve local consistency and inter-transaction consistency.

A novel Web services transaction dependency coordination protocol (WSTDCP) is proposed in this article. WSTDCP can effectively identify and resolve the transactions in an inconsistent state with dependency relations. Because of the property of isolation relaxation, dependency relations occur between concurrent transactions that access to shared resources. In the worst situation possible is that these dependencies may form a cycle, which cannot be seen locally by the providers. Therefore it needs to propose a mechanism to detect such cycles and to safely end the involved transactions without inconsistency.

Received date:23-08-2008
Corresponding author: WANG Xiao-jun, E-mail: xjwang@njupt.edu.cn
DOI: 10.1016/S1005-8885(08)60249-6

## 2 Related work

Several isolation-relaxing transaction models [2] have been introduced since late 1980s. Various protocols have been proposed for managing transactions in Web services including WS-CAF, WS-Business Activity, WS-Coordination, and business transaction protocol (BTP). These proposals do not handle the inconsistency problem of isolation-relaxing transactions. However, the problem of concurrency control was addressed by [1,3–4]. Alrifai M et al. [3] proposes an extension to the WS-Transaction Protocol which ensures the consistency of the data. Its extension avoids direct communication in transaction coordinators and thus preserves security. Choi S et al. [1] proposes a mechanism to ensure the consistent executions of isolation-relaxing WS transactions. The mechanism can discover a cycle by direct communication in transaction coordinators. Haller K et al. [4] introduces a new decentralized serialization graph testing protocol to ensure concurrency control and recovery in peer-to-peer environments.

## 3 Web services transaction model

WSTDCP is designed for the transactions based on the Web service business activity model. The underlying model of the protocol comprises a set of service providers that provide various Web services, and a set of service consumers which require a set of services $S=\{s_1,s_2,\ldots,s_n\}$. A composite Web service is composed of '$m$' component services ($1<m\leqslant n$) provided by service providers. A composite Web service is modeled as a composite service transaction (CST) $T$, and its component services $s_1,s_2,\ldots,s_m$ are modeled as component transactions $t_1,t_2,\ldots,t_m$. And a CST can define a set of permissible outcomes and must reach one of those outcomes when it ends.

### 3.1 Dependencies

Dependencies specify the compositional aspects of component transactions and how these transactions affect the behavior of other transactions during the process of execution of a composite service transaction [5]. Dependencies are enforced by defining various sets of rules.

**Definition 1** This completion dependency $t_b \succ t_a$ occurs between $t_b \in T_1$ and $t_a \in T_1$ when the successful end of $t_b$ is dependent upon the status of $t_a$ in the same CST $T_1$. Thus, $t_b$ cannot be closed before $t_a$ is closed. $t_a$ is referred to as the pre-ordered transaction of $t_b$, and $t_b$ as the post-ordered transaction of $t_a$.

**Definition 2** Let us assume that there are two component transactions $t_i \in T_1$ and $t_j \in T_2$ where both $T_1$ and $T_2$ are CST. The ended-state dependency $t_j \to t_i$ exists if the successful end of $t_j$ is dependent upon the status of a transaction $t_i$. $t_i$ is referred to as the pre-ordered transaction of $t_j$, and $t_j$ as the post-ordered transaction of $t_i$. This ended-state dependency is enforced by the following rules:

**Rule 1** Close rule. A post-ordered transaction must delay its successful end until all its pre-ordered transactions are closed.

**Rule 2** Compensation rule. If a pre-ordered transaction is compensated, all its the post-ordered transactions must be compensated.

For instance, the ended-state dependency $t_j \to t_i$ occurs if $t_i \in T_1$ and $t_j \in T_2$ access the same resource held by a common provider and $t_i$ influences the result of $t_j$. When a pre-ordered transaction $t_i$ is compensated, two solutions are obtained, redo and compensation, to handle its post-ordered transaction $t_j$. The redo scheme first compensates $t_j$, then re-executes it. Thus, it re-executes some individual transactions, rather than the entire CST. On the other hand, the compensation scheme only compensates the $t_j$.

**Definition 3** A local completion dependency graph (LCDG) is a directed graph<$N$,$E$>, where a vertex $a$ in $N$ represents a component transaction $t_a$ and an edge in $E$ from vertex $a$ to vertex $b$ indicates a completion dependency $t_a \succ t_b$. This dependency relation is transitive relation.

**Definition 4** A local ended-state dependency graph (LEDG) is a directed graph<$N$,$E$>, where a vertex $i$ in $N$ represents a component transaction $t_i$ and an edge in $E$ from vertex $i$ to vertex $j$ indicates an ended-state dependency $t_i \to t_j$.

**Definition 5** A global transaction dependency graph (GTDG) is a directed graph<$N$,$E$>, where a vertex $i$ in $N$ represents a component transaction $t_i$ and an edge in $E$ from vertex $i$ to vertex $j$ indicates an ended-state dependency $t_i \to t_j$ or a completion dependency $t_i \succ t_j$.

Fig. 1 shows an example of GTDG which includes four ended-state dependencies $t_1 \to t_2$, $t_3 \to t_4$, $t_5 \to t_6$, and $t_5 \to t_7$. In addition, each rectangular box respectively represents each CST's LCDG. For example, the composite service transaction $T_2$ has completion dependencies $t_4 \succ t_9 \succ t_5$. According to the definition of completion dependency, it is known that the successful end of $t_4$ is also dependent on the status of $t_5$. Thus, a reduced local completion dependency