A Reservation-Based Protocol for Timely Web Services Transaction

Lin Qing, Aziz Nasridinov, Jeongyong Byun
Department of Computer and Multimedia, Dongguk University
e-mail :qinglin9@gmail.com, aziz_nasridinov@yahoo.com, byunjy@dongguk.ac.kr

시간 요구 웹서비스 트랜잭션을 위한 예약기반 규약

림 청, 아지즈 나스리디노프, 변정용
동국대학교 컴퓨터멀티미디어학부

Abstract

Business transactional process is usually long running computation which requests services from multiple enterprises. Web Services Transaction specification (WS-TX) defines a protocol, WS-BusinessActivity which is specifically designed for such lengthy interaction and maintains overall consistency through compensation. However timely web services transaction is prone to compensate due to tasks' missing deadline. Therefore, this paper proposes a reservation-based protocol which is used to select providers who can provide resources promptly and deselect ones who may fail. And this selection happens during resource reservation phase and before real commitment. In this way, we achieve the goal of minimizing transactions' compensation. Finally, we design the framework architecture for the proposed protocol that is extended from WS-BusinessActivity.

1. Introduction

Web service-based business processes consist of long-running, complex transactions involving numerous services. However it's impossible to utilize ACID in business processes, for the reason that exclusive locking resources over extended periods of time is impractical and the isolation is relaxed, though we still hope for all-or-nothing semantics, consistent outcomes and some level of durability [1]. Fortunately, candidates for just such a protocol do exist, which is Web Services Transaction specification (WS-TX). It consists of WS-Coordinator, WS- AtomicTransactions and WS-BusinessActivity. WS-BusinessActivity is a protocol, which specifically designed for such lengthy interaction and maintains overall consistency through compensation.

However, timely web services transaction is prone to compensate due to tasks' missing deadline. For example, consider a Web service-based process for ordering two car parts from two retailers. Both two parts are indispensable to repair a car and customer has a deadline. That means if one task in the ordering transaction is late, customer won't be interested in a successful termination that violates the deadline. And then the transaction has to be compensated.

In this paper, we ensure transactional timeliness through two aspects. One is to select providers that can give resources promptly. And it's better to do this before transaction's real commitment because compensation is not expected. Hence our solution is to apply resource reservation with a time constrain. We say only by every task's gaining resource reservation in time can the transaction begins to commit. Otherwise it has to be cancelled, even if only one task fails to get reservation in time. This is required by transactional properties such as atomicity and consistency.

The other important aspect to ensure timeliness is that deselecting unconfident providers during resource reservation phase. "Unconfident" means even reservation is obtained by a task, resource provider has potential problems that prevent transaction from completing on time. The most crucial one is concurrent conflict in provider's side. For example, ORDER transaction depends on SUPPLY transaction when the resources are insufficient for ORDER. And two business transactions are concurrently processed. In this case, ORDER has to wait for SUPPLY's completion. [4] found a solution to detect and to ask transaction to wait. Therefore, our protocol is designed to avoid that kind of concurrent conflict through resorting to alternatives during resource reservation phase.

In this paper, we extend WS-BusinessActivity protocol [4] to ensure every task in one transaction to get resource reservation in time so that the transaction can process without delay. In addition, providers' time limit for resource reservation is taken into consideration. Finally, we achieve a goal of minimizing the number of compensated transactions due to missing deadlines.

The rest of paper is organized as following: in the 2nd chapter we will discuss related studies. Then we will focus on motivating scenario and discuss proposed method. System design and algorithm will be illustrated in the 5th chapter. Paper will be concluded with conclusion.

2. Related Studies

[5] describes a reservation-based extended transaction protocol that can be used to coordinate such business activities. However, they didn't consider time constrains for reservation and their protocol doesn't work when it encounters concurrent conflict. That means delay still exists. Moreover, system design isn't given out in this paper. Therefore, our paper is going to enhance their solution and make a design extended from WS-Tx.

[6] paper's aim is to reach an agreement between the client and all participating providers (participants) on what transaction processing times have to be expected, accepted and guaranteed. In this way, a global consistency and correctness can be guaranteed. Actually applications can't control the moment of resource locking in that time span. Because locking database is a problem left to internal database control. It's impractical to apply this mechanism in reality. Hence we propose to use resource reservation, which can be fully controlled by web service application.

[7] solved the concurrent problem of web service transaction through provider sides' serialization graph which contains transactional dependency information. And "wait" state is added into their protocol to avoid conflict. Obviously, since that waiting time is imponderable due to human involvement, time constrain should be taken into account when applying to timely web service transactions. It is the problem that we want to solve in our paper.

[8] tries to select the most proper providers who give reservation for web service transaction. Since their selection criteria includes customer's deadline, they have to evaluate providers' processing time. It is hard to do because of business processes' dynamicity. In this paper, we select the most appropriate one depending on participants' ability of resource reservation before the pre-designed deadline.

3. Motivating Scenario

The following scenario in Figure 1 shows an order transaction and its hierarchy model. Here we have one transaction called "Car parts order" and its two tasks "Transmission Order" and "Braking Order".

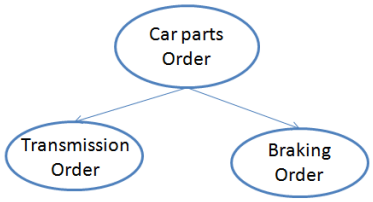

Figure 1.The example of an order transaction and its hierarchy model

Let's suppose someone's car gets crashed and two parts, i.e. transmission and braking, need to be changed. The repair shop's application can be used to order them automatically from two retailers by connecting to their web services. Particularly customer wants his car to be repaired in two days. Otherwise he will give up repair. In this case, repairman says he needs one day to fix the car, so all the parts are supposed to arrive on the first day.

Problems come when braking retailer notifies repair shop a delay. While it's already in the late afternoon, new transmission has arrived, and it's too late for delivery. Consequently, Transmission order task has to compensate. This is a typical problem caused by overtime tasks in a timely business transaction.

Our solution is to divide the order task into two phases: reservation and order, and give a deadline for reservation phase which is shown in Figure 2. From this figure, we also can tell how this reservation phase behaviors and how important it is for the whole car repair activity.

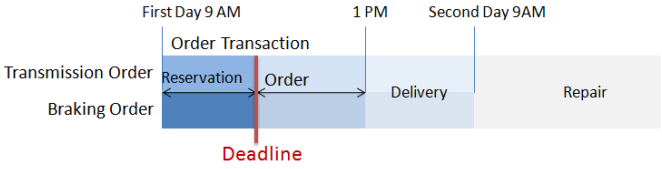

Figure 2. A solution proposed to solve the late problem

During reservation phase, resource reservation results are collected and evaluated. Eventually transaction chooses participants for each task. After actual resource reservation, customer can complete and pay at any time but before retailer's reservation deadline. Also this resource is no longer accessible for transactions. The order phase involves the completion of the transaction.

4. System Design Extended from WS-TX

In this part, we are going to apply our solution to WS-Tx. This transaction specification defines mechanisms for transactional interoperability between Web services domains and provides a means to compose transactional qualities of service into Web services applications [2].

Here we assume there are always plenteous resource retailers. Since our main purpose is to apply reservation mechanism to WS-Tx, WS-BusinessActivity's compensation part won't be repeated in this paper. The detailed description is in [4], WS-BusinessAcitiviy specification.

Figure 3 is the coordinator's framework which is extended from WS-Coordinator [3]. The grey arrows denote message communications, which reference to WS-Coordinator [3] directly. And the deep blue arrows are the extended parts. Besides, there resides Local Serialization Graph [7] in provider's side, which is used to record dependent relationship with other transactions.

Figure 4 shows the sequence diagram which depicts every possible interaction in our design. And the deep blue arrows point out the interactions we design for resource reservation. Moreover, the corresponding interaction modules in the framework can be found in Figure 3 the blue strips in Figure 4 mean that time constrains have effect on interactions.

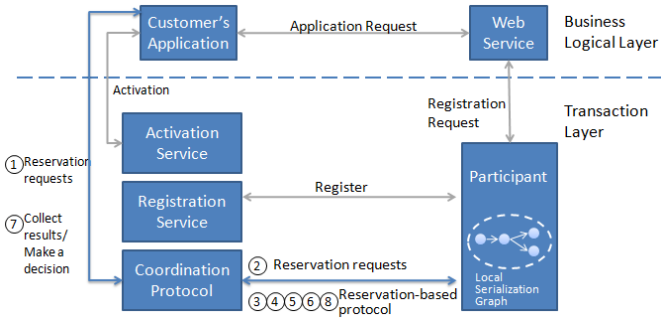

Figure 3. Extended Coordinator Framework and Message Flow

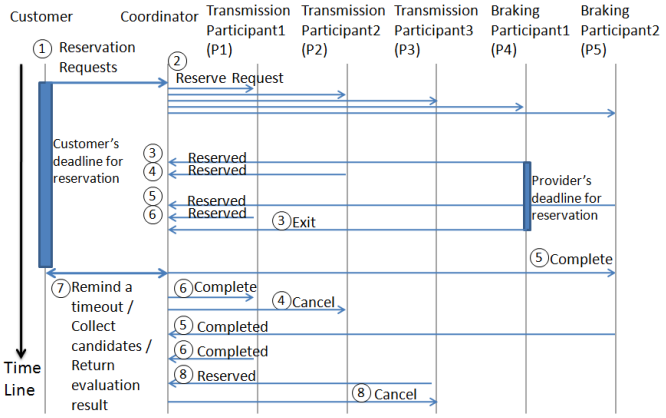

Figure 4. Sequence Diagram for our protocol extended from WS-BinessActivity

Step1: Customer's application sends message to coordinator to ask for reservation.
Step2: Coordinator sends reservation requests to all of the participants.
Step3: Braking Participant1 (P4) gives the reservation, but P4 has a time limit for customer's holding resource. So when time is out, P4 has right to exit the transaction.
Step4: The coordinator gets the reserved reply from Transmission Participant2 (P2), but with a notification saying there is a concurrent conflict with other transactions in P2's side. As long as a more proper candidate, e.g. Transmission Participant1 (P1), appears, the coordinator cancels this task from P2.
Step5: A successful reservation from Braking Participant2 (P5).
Step6: A successful reservation from Transmission Participant1 (P1).
Step7: When the time for resource reservation runs out, customer's application collects candidates from coordinator and decides which one to choose for each task.
Step8: Transmission Participant3 (P3) gives the reservation overtime, and coordinator cancels it directly.

One important thing we have to point out is if no proper candidate appears for one task, and if this task is absolutely essential to the transaction it belongs to, the coordinator can repeat the reservation collection until it gains all necessary reservations. Otherwise the transaction should close when to reach a customers' maximum waiting limitation.

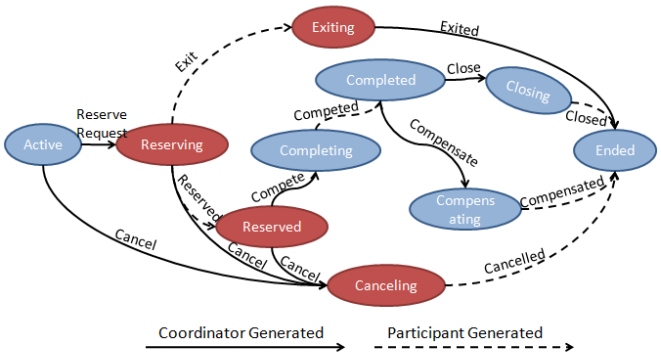

Figure 5. State Diagram

Figure 5 shows the participants' state and message flow. The red circles are the states we add to the original WS-BusinessActivity protocol.
Reserving: It is a state, which is after participants' registering into the coordinator and receiving Reserve Request message, when participants are making reservation for this transaction.
Reserved: Resource is reserved by this transaction, and others can't access to this part of the resource anymore. Once customer wants the participants to complete, he can get the resource easily.
Canceling: It is for canceling and removing participants from this transaction. This canceling is much easier than compensation, which is also a kind of cancel, but would generate more cost and can't eliminate effects entirely.
Exiting: It is the state for participants' quit from this transaction autonomously. Especially when web services providers don't allow transaction's long holding resource.

5. Algorithm Design

In this part, we explain the algorithms which can be applied to coordinator and customer's application.

Definition:
Dc = customer's deadline for reservation resource
$T = \{T1, T2, \ldots, Ti, \ldots Tn\}$ is a set of tasks in one transaction
$P = \{P1, P2, \ldots, Pk\}$ is a set of participants who give reservation, briefly called candidates in this paper.
During the resource reservation collection period, F list is formed.
$F = \{F1, F2, \ldots, Fi, \ldots Fn\}$ list of candidates for task Ti respectively.
$F1 = \{ P1, P2, \ldots Pm\}$ $(0<m<=k)$
…
$Fi = \{ P3, \ldots Pm\}$ $(0<m<=k)$ is a set of candidates for task Ti.
…
$Fn = \{ Pm \ldots \}$ $(0<m<=k)$
n denotes the number of the tasks in one transaction
k denotes the number of candidates for all the tasks
m denotes the number of candidates for task Ti

During the resource reservation phase, the coordinator will collect result about which candidate for which task. And Algorithm 1 can be applied to the coordinator's module.

Algorithm 1: Resource Reservation Phase

Input:
Dc = deadline for reservation resource
T = {*T1, T2, …, Ti, … Tn*} is a set of tasks in one transaction

Start:
While (NowTime–StartReservationRequestTime < *Dc*)
Foreach *Ti* in *T*
If (Message==“*Reserved*” & ResoureNo== *i*)
Push *Pk* into *Fi*
k++
ElseIf (Massage == “*Exit*” & ResourceNo == *i*)
Delete *Pk* from *Fi*
k++
EndIf
EndForeach
EndWhile

It's time to select the right Participants from those candidates and decide the transaction to complete or not according to *F* list. Algorithm 2 has the responsibility to do this job and it is applied into customer's application in the business logic layer.

Algorithm2: Participant Selection from Candidates

Input:
F = {*F1, F2, …, Fi, … Fn*} list of candidates for task *Ti* respectively
Fi = { … *Pm*…} (*0*<*m*<*k*) is a set of candidates for task *Ti*

Start:
Foreach *Fi* in *F*
If (*Fi* != NULL)

Select one Participant *Pm* according to business rules (e.g. select the cheapest one and deselect ones that concurrent conflict with other transactions exists)

Inform Coordinator to send *Pm* “Complete” message

Inform Coordinator to send other candidates “Cancel” messages

ElseIf
If (*Ti* is necessary in *T*) /*Check business rules*/
Inform Coordinator to send every *Pm* in *P* “Cancel” message
ElseIf
Continue
EndIf
EndIf
EndForeach

6. Conclusion and Future Work

In this paper, we propose a transaction framework extended from WS-Tx, which is specifically designed for timely web service transaction. In order to solve the late problem of tasks in a transaction, we raise our idea of resource reservation with a deadline. In addition, the providers who are suffering from concurrent conflict can be perceived by the customer during reservation phase. So that transaction can avoid waiting and delay. Finally, we achieve the goal of minimizing the number of compensated transactions due to missing deadlines.

However, we don't consider about nested web service transaction. And we simply take the top transaction for our research and the deadline is given by customer. In the future, we will work on adaptively scheduling for business transaction including all or part of its nested sub-transactions.

Reference

[1] Sandeep Chatterjee, James Webber, Developing Enterprise Web Services: An Architect's Guide. Prentice Hall PTR, 2003, pp. 272.

[2] IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA,” Web Services Transactions specifications”, http://www.ibm.com/developerworks/library/specification/ws-tx/.

[3] *OASIS Web Service Coordination (WS-Coordination)* Version 1.2, http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.pdf, 2009.

[4] *OASIS Web Service Business Activity (WS-BusinessActivity),* http://docs.oasis-open.org/ws-tx/wsba/2006/06, 2009.

[5] Wenbing Zhao, L. E. Moser, P. M. Melliar-Smith, "A Reservation-Based Coordination Protocol for Web Services," icws, pp.49-56, IEEE International Conference on Web Services (ICWS'05), 2005

[6] M. Alrifai, W.-T. Balke, P. Dolog, and W. Nejdl, “Non-Blocking Scheduling of Web Service Transactions,” *Proc. European Conf. Web Services (ECOWS '07),*2007.

[7] Mohammad Alrifai, Peter Dolog, Wolf-Tilo Balke, Wolfgang Nejdl, "Distributed Management of Concurrent Web Service Transactions," IEEE Transactions on Services Computing, vol. 2, no. 4, pp. 289-302, Oct.-Dec. 2009, doi:10.1109/TSC.2009.29

[8] Aziz Nasridinov, KyongWook Kim and JeongYong Byun. “A Semantic Services Selection with Functional-Level Mediator in SOA”. Proceeding of International Conference on Semantic Web and Web Services, SWWS2009, WORLDCOMP09’, Las Vegas, Nevada, US.