A Generic Architecture for SOAP Transaction Management

Lyndal Kanagasabai[1,2], Wayne Brookes[1]

[1]University of Technology, Sydney,
PO Box 123, Broadway NSW 2007, Australia
[2]Insession Technologies
Level 13, 234 George Street, Sydney NSW 2000, Australia
Lyndal.J.Kanagasabai@uts.edu.au, brookes@it.uts.edu.au

Abstract. Web Services and SOAP, the Simple Object Access Protocol, looks set to become not only the cornerstone of transactions between businesses, but SOAP is also is an emerging protocol for application interoperability within organisations. A successful, standard, transaction management protocol for SOAP is vital to its ongoing success, yet there is no single protocol standard emerging. A generic transaction management architecture is proposed to solve the problem of conflicting standards by concurrently supporting Web Service partners which use different transaction management protocols. A prototype of such an architecture was developed, for both the BTP and TIP protocols, showing that not only is the proposal feasible, but that it can meet all of SOAP's distributed transaction management requirements, without the need to wait for a suitable, ubiquitous protocol to emerge.

1 Introduction

Web Services are emerging as a key paradigm for supporting application interoperability, primarily in a business-to-business (B2B) environment. Web Services are based upon three core standards: SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) and UDDI (Universal Distribution, Discovery and Interoperability). However, these three basic Web Services standards do not address all of the necessary requirements for successful application interoperability. This paper focuses on one of the requirements that is lacking – transaction management.

An electronic transaction is a well-established concept. It represents a set of basic operations that must be treated as a single, indivisible activity to the user. Traditionally, a transaction requires the ACID properties of atomicity, consistency and isolation from other transactions, as well as being able to produce durable results.

Web Services are a unique concept in distributed transaction processing, because they are intended for B2B communication between partners, without requiring a trust relationship or prior arrangement. It is generally agreed that traditional distributed transaction protocols, with an emphasis on strict ACID properties, are not suitable,

because they require resources to be blocked for the duration of a transaction, something that is not appropriate in an Internet environment.

However, the machine-independent nature of the Web Services protocols makes them not only useful in a B2B scenario, but also for performing distributed transaction processing within an organisation's intranet. Web Services in an intranet environment may take one of two forms. In the simplest case, SOAP can be used as a protocol to expose the existing interfaces of legacy applications as Web Services, without fundamentally changing the applications themselves. Or in a more complex scenario, Web Services can be used to enable Enterprise Application Integration (EAI), by supporting interactions between existing heterogeneous systems.

Here we characterise the use of Web Services into three different application scenarios, each with differing transactional requirements:

- **Legacy.** In the more conservative legacy scenario, full ACID properties of transactions normally must be preserved. The transaction will still occur within a trust boundary, application logic remains the same, and hence full atomicity, consistency, isolation and durability are required.
- **B2B.** For the other extreme, B2B, allowances must be made for long-running transactions, so a "relaxed" version of ACID transaction management is required. Transactions occur across trust boundaries (between organisations), and in a less secure environment, so locking of transactional resources is not advisable.
- **EAI.** Between these two lies the EAI scenario, in which transactions may be short or longer-lived, and depending on the organisation's structure, trust boundaries may or may not be crossed. Hence some mixture of support for ACID and relaxed ACID transactions is a likely requirement.

The industry has not yet reached consensus on a protocol for Web Services transaction management. Three important industry groups, to address the varied requirements, have proposed three different protocols: TIP (Transaction Internet Protocol), BTP (Business Transaction Protocol) and the combination of WS-Coordination and WS-Transaction.

Although each of these protocols attempts to address some or all of the transaction management requirements, they are not compatible with each other. The last protocol pair, WS-Coordination and WS-Transaction, has only recently been proposed, and is not specified to an extent suitable for implementation. At this stage, it is not possible to predict which protocol, if any, will eventually attain broad acceptance, although BTP and WS-Coordination/Transaction appear to be the more likely candidates. Hence any vendor developing Web Services software is left in a quandary as to the most appropriate protocol to implement.

This paper outlines a generic architecture for Web Services transaction management, which can be used in a B2B, EAI or simple legacy application scenario, using SOAP as the Web Services transport protocol. The architecture is intended to be suitable for concurrent implementation and use of any or all of the transaction management protocols mentioned above, and hopefully, any future protocols that may be introduced for distributed transaction management.

This paper first introduces the different transaction management protocols for SOAP, and discusses their suitability for the different usage scenarios (Legacy, B2B, EAI). Then the proposed generic architecture is introduced, followed by an overview of a prototype implementation demonstrating the feasibility of the approach.

2 Protocols for SOAP Transaction Management

There are currently three candidate protocols for Web Services transaction management: TIP, BTP and WS-Coordination/WS-Transaction as outlined below.

2.1 Transaction Internet Protocol

The Transaction Internet Protocol 3.0 (TIP) is defined in an RFC [1], and currently has IETF "Proposed Standard" status. TIP dates from before the first proposals for SOAP and Web Services, and is both application and transport protocol independent.

TIP was designed with the ACID principles of transactions in mind, and consequently does not consider the case of long-running transactions.

Although TIP was originally intended for communication between traditional transaction managers, it has been suggested for use between a client application and transaction servers. Indeed, in 2001, Keith Evans, one of the TIP authors, suggested that TIP is ideal for "web agency type applications, which act as brokers for the services of other providers." [2]. Vogler, et al. [3] also suggest use of TIP for WWW applications, and describe an example implementation. Both papers however, still focus on full ACID transaction properties and do not acknowledge any WWW need for supporting long-running transactions and autonomous organisations.

TIP could be a suitable approach for transaction management wherever SOAP is used as a direct replacement for a legacy front-end, and for EAI scenarios where long-running transactions are not required. Also, without some modification of the protocol to allow it to be used directly between transaction participants, trust and security issues mean that TIP is unlikely to be popular where autonomous parties' transaction managers must communicate directly.

2.2 Business Transaction Protocol

The Business Transaction Protocol (BTP) Version 1.0 [4] was developed by the OASIS Business Transactions Technical Committee. Unlike TIP, BTP messages are XML-based. A SOAP binding is defined (for carrying BTP messages), however BTP could be used with any application message protocol.

BTP's major design goal is to cater for loosely coupled, distributed, transaction applications between autonomous organisations without assuming strict ACID principles of transactions.

BTP transactions are categorised as either atoms or cohesions. An atom is a transaction with all the ACID principles, except it relaxes the isolation principle [5]. A cohesion also relaxes ACID, by allowing the participants to negotiate the atomic properties of the transaction. In a cohesion, consistency is maintained, the isolation principle is relaxed as with atoms, and durability is also relaxed [5].

The BTP protocol cannot achieve full ACID, due partly to its semantics and partly to the heuristics used with its commit protocol.

The focus of all BTP development and discussion in the industry is towards true B2B Web Services and solving the problem of managing long-running transactions

between autonomous participants. The BTP specification appears to accomplish this goal by allowing transaction participants to themselves determine the degree of ACIDity they will or must support, including a mechanism for dynamic negotiation, but fails to explicitly cater for transactions with full ACID.

2.3 WS-Coordination and WS-Transaction

WS-Coordination and WS-Transaction are sister specifications for Web Services transaction management, published by IBM, Microsoft and BEA in [6] and [7].

WS-Transaction defines the protocol sets that can be used for transaction management and includes options for both atomic and long-running transactions. WS-Coordination is described in its specification as "an extensible framework for providing protocols that coordinate the actions of distributed applications." It could also be defined as an XML-message-based API for transaction participants to create transaction contexts and to register for the various transaction management protocol options provided by WS-Transaction.

WS-Coordination and WS-Transaction provide support for SOAP's major transactional requirements by supporting both ACID and long-running transactions.

The WS-Coordination and WS-Transaction specifications have been published for "review and evaluation only." As such, they are not rigorous specifications, omitting some details such as error recovery procedures and message formats (other than in external schemas). Therefore, the specifications are not at a stage where a third party could accurately implement them - an implementation could only be built from these specifications by making assumptions about the missing details.

However, from the details that have been published, it appears that these protocols will be suitable to all of the scenarios where SOAP could be used, in legacy, B2B and EAI scenarios, due to their support for both atomic and long-running transactions.

2.4 Summary of Candidate Protocols

Of the three protocols presented, WS-Coordination/Transaction is promising but still immature, therefore at the present time, BTP and TIP are left as the two most suitable contenders for SOAP transaction management.

However, considering the legacy, B2B and EAI scenarios, neither protocol completely meets the requirements of each. BTP caters only for relaxed isolation and atomic principles, which is not suitable for legacy scenarios. TIP provides full ACID support, but does not directly satisfy the requirements of long-running transactions.

The most ideal implementation would use a combination of both TIP and BTP and be extensible to WS-Coordination/Transaction. This is possible given the similarities in architecture between the three protocols. An additional benefit of such a solution is that the resulting implementation would be compatible with solutions from multiple vendors, regardless of which of these three transaction management protocols they eventually choose for use in their products.

3 A Generic Transaction Gateway

Given that there is no single transaction management protocol that fulfils all of the requirements of different applications of Web Services, this paper proposes a gateway approach to allow SOAP applications to simultaneously support multiple transaction management protocols using a single, protocol-agnostic API. Further, it theoretically means that any single, nested transaction may in reality be implemented with different transaction protocols between the various transaction participants.

Analysis of the transaction management protocols has shown several areas of commonality between them that make the concept of a gateway architecture possible. The most fundamental aspect of commonality is the basic model of interaction: in all cases, applications "talk" to a transaction coordinator that implements the protocol. The second area of commonality is the APIs used between applications and the local transaction coordinator. In TIP, BTP and WS-Coordination/Transaction, the APIs are roughly equivalent in terms of abstract function, but the abstract functions are known by different names in each protocol. In essence, the most significant difference between the transaction management protocols is the "across the wire" messages and states between transactional nodes.

After examining the currently available transaction management protocols for SOAP, a generic transaction gateway would need to meet the following requirements:

- Concurrently support multiple transaction management protocols, and interoperate with any standard-conforming implementation of each protocol it supports;
- TIP and BTP must be supported at a minimum;
- Be extensible for any future protocols that may become prominent, especially WS-Transaction and WS-Coordination;
- Be portable across different platforms and able to be integrated into different transaction processing architectures;
- Allow a node to simultaneously take the role of transaction participant and/or coordinator;
- Support the normal architectural requirements of distributed transaction processing systems, such as scalability, reliability, short response times, high availability and cost-efficiency;
- Be compatible with SOAP when applied to legacy, B2B and EAI scenarios.

3.1 Transaction Gateway Architecture

At a high level, the architecture consists of two main parts, as shown in Fig 1:

- Transaction Gateway API (TGAPI) that provides an interface for applications using the Gateway;
- The Transaction Gateway component itself.

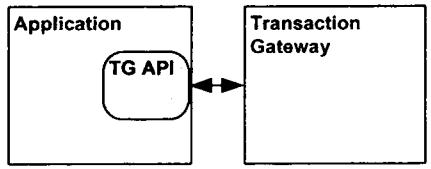

Fig. 1. Transaction gateway relationships

The **Transaction Gateway API** provides a consistent interface for programmatic users (applications and/or a SOAP service component and/or a higher level wrapper API) to initiate, control or participate in distributed transactions, independent of the transaction management protocol being used. It provides the means for the user to choose the transaction management protocol that will apply to the transaction for its duration, if the user is a transaction coordinator (client).

The **Transaction Gateway** component bridges in both directions between requests and responses delivered by the API and the transaction management protocol that is being used for a particular transaction. It implements the finite state machines of the protocols it supports, implements persistence where required and performs other tasks as required for conformance with the appropriate specifications and standards. It is responsible for interfacing to the lower layer protocols (SOAP, HTTP, TCP, etc.) and uses the services of the local transaction manager provided by the operating system to start, stop and abort transactions locally, on behalf of the application.

Internally, the Transaction Gateway consists of sub-components for processing API requests and sending API responses, for implementing the various transaction protocols, and for interfacing to lower layer transport networks, as shown in Fig 2.

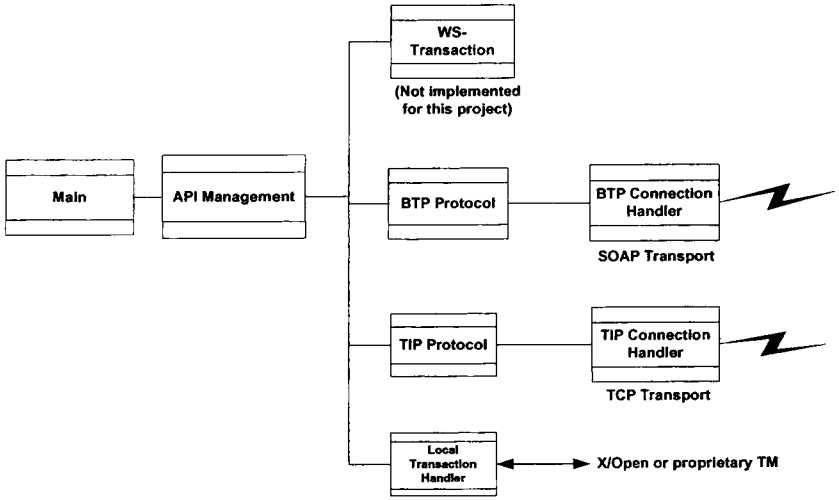

Fig. 2. Transaction gateway internal structure

3.2 Prototype and Results

For this project, a prototype of the Transaction Gateway architecture was implemented, as a proof of concept. Both the Transaction Gateway API (used by clients) and a Transaction Gateway itself were implemented.

Only limited transaction management functionality was implemented for this project, but sufficient to demonstrate the architecture's feasibility, namely:

- Begin a TIP "push" transaction;
- Begin a BTP transaction;
- Begin a BTP transaction with context.

These functions included implementation of the applicable protocol-level messages required to facilitate these, e.g. TIP PUSH command and BTP ENROL command.

The tests conducted demonstrate that the Transaction Gateway architecture meets the following requirements:

- TIP and BTP can be concurrently supported, and the architecture is theoretically extensible to WS-Coordination/Transaction;
- It is possible to create a generic API that encompasses functions from the different transaction management protocols (and supporting both ACID and relaxed ACID requirements);
- Allows a node to act in transaction participant and/or coordinator roles;
- Is compatible with SOAP;
- Supports geographically distant nodes.

Most importantly, the gateway allows a client (the transaction coordinator) to begin a transaction using any of the transaction management protocols supported by the gateway merely by specifying the required protocol at the time the connection to the gateway is opened. The same API for creating and managing transactions is used regardless of which protocol is selected, reducing both the complexity of the client code and removing the need to install protocol-specific libraries on the client. A single client can even participate in transactions using multiple protocols simultaneously, by opening multiple connections to the gateway.

4 Conclusion

This paper has investigated the field of transaction management, from the viewpoint of business-to-business, legacy and EAI applications of the SOAP protocol.

It has found that all three types of SOAP applications have much in common with traditional management of distributed transactions. However, the business-to-business nature of Web Services, and to some extent, EAI applications, is likely to require relaxed ACID semantics, in order to accommodate long lived transactions, an unstable environment and the spanning of trust boundaries by transactions.

Although the theoretical basis for addressing these issues is well-established, new, standardised, transaction management protocols are required to apply this theory, as well as support the traditional full-ACID properties for legacy applications, so that heterogeneous transaction management environments can interoperate.

There has been significant industry activity towards various aspects of this goal. Three distributed transaction management protocols have received the main industry focus: TIP, an IETF protocol, BTP, proposed by the OASIS consortium and WS-Coordination/Transaction which was put forward by BEA, Microsoft and IBM.

Despite these efforts, no technology yet appears to be emerging to become the ubiquitous solution that is required. It leaves the industry in a situation where the interoperability promises of SOAP and its associated standards are compromised due to the lack of a single distributed transaction management protocol that can be relied upon as a standard, de facto or otherwise.

To address this interoperability problem for distributed SOAP transaction management, this paper proposes a generic transaction management architecture as a solution. The intention is to work around the problem of conflicting standards, by allowing an application to participate in transactions with other applications, even if multiple heterogeneous transaction management protocols are used by those other applications. The Transaction Gateway approach is designed to concurrently support any or all distributed transaction management protocols that may conceivably be required for SOAP, yet provide a single, protocol-independent API for use by transaction participants.

The design, development and prototype development of the proposed Transaction Gateway, and the subsequent tests run against it, have demonstrated that it is a viable concept. It is capable of using existing infrastructure facilities for communications, and most importantly multiple distributed transaction management protocols can co-exist in the one product, sharing a common API.

References

1. Lyon, J., Evans, K., Klein, J.: Transaction Internet Protocol Version 3.0. Internet Engineering Task Force RFC 2371 (1998)
2. Evans, K.: Transaction Internet Protocol: Facilitating Distributed Internet Applications. In: W3C Workshop on Web Services, San Jose, CA, USA (2001)
3. Vogler, H., Moschgath, M. L., Kunkelmann, T., Grunewald, J.: The Transaction Internet Protocol in practice: reliability for WWW applications. In: Internet Workshop '99 (IWS'99) (1999) 189-194
4. Ceponkus, A., Dalal, S., Fletcher, T., Furniss, P., Green, A., Pope, B.: Business Transaction Protocol [Online]. Organization for the Advancement of Structured Information Systems (2002) Available: http://www.oasis-open.org/committees/download.php/1184/2002-06-03.BTP_cttee_spec_1.0.pdf [Accessed 12 Dec 2003]
5. Potts, M., Cox, B., Pope, B.: Business Transaction Protocol Primer [Online]. Organization for the Advancement of Structured Information Systems (2002) Available: http://www.oasis-open.org/committees/business-transactions/documents/primer/BTP_Primer_v1.0.20020603.pdf [Accessed 12 Dec 2003]
6. Cabrera, L. F., Copeland, G., Cox, W., Feingold, M., Freund, T., Johnson, J., Kaler, C., Klein, J., Langworthy, D., Nadalin, A., Orchard, D., Robinson, I., Shewchuk, J., Storey, T.: Web Services Coordination (WS-Coordination) [Online]. IBM developerWorks (2003) Available: http://www.ibm.com/developerworks/library/ws-coor/ [Accessed 12 Dec 2003]
7. Cabrera, F., Copeland, G., Cox, B., Freund, T., Klein, J., Storey, T., Thatte, S.: Web Services Transaction (WS-Transaction) [Online]. IBM developerWorks (2002) Available: http://www.ibm.com/developerworks/library/ws-transpec/ [Accessed 12 Dec 2003]

Savitri Bevinakoppa and
Jiankun Hu (Eds.)

Web Services: Modeling, Architecture and Infrastructure

Proceedings of the
2nd International Workshop on
Web Services: Modeling, Architecture and Infrastructure
WSMAI 2004
In conjunction with ICEIS 2004
Porto, Portugal, April 2004

INSTICC PRESS
Portugal

Workshop Chairs

Savitri Bevinakoppa (savitri@cs.rmit.edu.au)
Royal Melbourne Institute of Technology
Australia

And

Jiankun Hu (jiankun.hu@rmit.edu.au)
Royale Melbourne Institut of Technology
Australia

Program Committee

Albert Y. Zomaya, CISCO, USYD (Australia)
Alex Delis, Polytec University (USA)
Boualem Benatallah, University of New South Wales (Australia)
Jean-Jacques Moreau, Canon (France)
Jen-Yao Chung, IBM (USA)
Jorge Cardoso, University of Georgia (USA)
Steve Vinoski, IONA (USA)
Sumi Helal, University of Florida (USA)
Yetongnon Kokou, Université de Bourgogne (France)
Jin Song DONG, NUS (Singapore)
José Miguel Baptista Nunes, Sheffield (UK)

Table of Contents

Invited Speakers

Full Papers

Posters