

On Generating Dependable Decentralised Web Service Business Sessions

Dacheng Zhang and Jie Xu
School of Computing, University of Leeds
Leeds LS2 9JT, England
{dcz, jxu@comp.leeds.ac.uk}

Abstract

Current solutions for generating Web service business processes (e.g. BPEL4WS) are generally based on centralised models, where a business process is driven by a centralised workflow engine, which schedules the work to be done and assigns it to an appropriate executor. Consequently, current solutions suffer from drawbacks (e.g., poor scalability, vulnerability to failures, etc.), caused by inadequate adoption of centralised models. There are some projects aiming to address these drawbacks from a system architectural point of view, and as a result, various decentralised solutions [2, 4] have been proposed. However, generating decentralised Web service business processes is still a new research topic, and a lot of issues are not yet well understood. The focus of this paper is to discuss two fundamental requirements, Web service instance level authentication, and heterogeneity in exception handling, for generating dependable decentralised Web service business process executions (business sessions).

1. Introduction

The BPEL4WS specification allows people to specify Web service business processes in a centralised model. BPEL4WS assumes that the entire specification of a business process is executed on a centralised coordinator, and all the messages amongst the business partners are transferred through and processed by the coordinator. The advantages of the centralised solution are that workflow execution is fully controlled by a WFMS. Communication between the coordinator and the nodes executing activities is relatively simple (in a star model). However, as all communication is passed by the coordinator, the communication overload is relatively high; and in some cases, the performance of the centralised coordinator will become a bottleneck and limit the scalability of the system. There are also arguments that this model is not flexible enough for scenarios where data flow has to be transported in a given direction due to certain business constraints [4]. Aiming at overcoming these disadvantages, decentralised solutions [2, 4] have been proposed. In these solutions, the specification of a business process is executed on diverse nodes, and communication among the business partners is in a peer-to-peer manner. The decentralised solution may lead to increased parallelism and reduced message overhead, since data flows and control message flows can be transported along dependence edges. In the decentralised solution, data flows and control flows can be very complex, as business partners communicate in a peer-to-peer

manner. Furthermore, business logic may execute on different nodes simultaneously. This introduces new challenges to business flow management. In this paper, we will focus on two issues; Web service instance level authentication, and heterogeneity in exception handling. These issues are critical for generating dependable decentralised Web service business process executions (business sessions). In Section 2, different solutions to Web service instance identification and their application areas are discussed. Section 3 presents our authentication mechanism at service instance level and the results of its experimental evaluation. Section 4 introduces a new abstract exception hierarchy. In Section 5, our conclusion is provided.

2. Service instance identification

A Web service business session is made up of multiple Web service instances (session partners). As mentioned above, the data and control flows in a decentralised business session can be very complex. On some occasions, a session partner may even have to deal with messages from other session partners that are unknown to it in advance. Therefore, it is necessary to generate a Web service instance identification mechanism for session partners so that every message can be dispatched to the appropriate Web service instance and the recipient of the message can verify where the message comes from.

Generally, there are two categories of schemes for service instance identification, namely the

ID-based solution and the Token-based solution. The ID-based solution is an attempt to apply a remote reference idiom to Web service design [3]. In this solution, Web service instances are explicitly identified with instance identifiers. Consequently, business flow management system can have details of partner instances and manage them in a precise manner. The Token-based solution uses the correlation information to identify a conversation amongst service instances and then implicitly identify the instances involved [1]. This solution has been leveraged in the BPEL4WS language [1] to help locate the component service instances within a composite Web service instance. The two categories of solutions are suitable for different application requirements although they can be replaceable in some scenarios. The Token-based solution can be employed to generate relatively coarse-grained solutions, for instance, a shared token can be used to identify a group of principles with some kind of trust relationship, which cannot be achieved by the ID-based solution. On the contrary to the Token-based solution, the ID-based solution suits for the scenarios where the identity of every principal needs to be specified.

In the next section, we will introduce our multi-party authentication system. Our solution realizes authentication at the instance level and leverages the ID-based solution to identify the instances within a business session.

3. Instance level authentication

In practice, communication among session partners in some aspects is controlled by session partners themselves rather than infrastructure protocols; that is, there are many occasions when a session partner should be able to decide the session partner it intends to contact according to the business specification. Therefore in some attacks, adversaries may be able to control a service instance to send a message to an un-intended recipient instance by, for example, forwarding incorrect routing information to it. In order to address this security issue, we have designed an instance level authentication mechanism for session partners within the same business session to help them establish a reasonable level of trust. This mechanism is an ID based authentication system. In this system, every session partner is assigned a unique identifier, and a corresponding private key. In conversations, a session partner can use the identifier to indicate its identity and use the private key to prove the

possession of the identifier. As illustrated in Figure 1, in our authentication system there is a Session Authority (SA) component, which takes charge of distributing session authentication related messages and provides reliable real-time information about a session to its session partners. Each experimental Web service is associated with a session handler and a session service. Session handlers are responsible for inserting the instance identification information into the headers of the SOAP messages, locating the service instance according to the instance identifier, and contacting with the Session Authority. Specific services called session services are used to generate the identifiers and the corresponding private keys for newly generated instances. A requestor can contact the session service associated with the invoked service and obtain the identifier of a new instance to be invoked before it sends the invoking message to the invoked Web service. It is particularly useful in asynchronous scenarios, where a session partner may invoke another Web service and access the new invoked new instance without requiring any reply. In this case, the session partner can obtain the identifier of the instance from the session service and introduces the instance to the SA before it sends off the invoking message. It can thus be sure that the newly invoked instance has been accepted as a session partner before it attempts to introduce other Web service instances into the business session. Basically, the instance identity verification is integrated within infrastructure protocols. Details about this authentication system are introduced in [7].

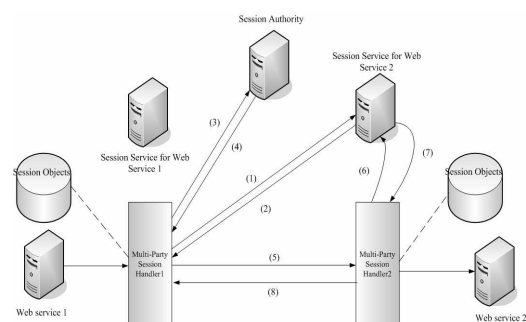


Figure 1. Authentication process

In order to evaluate the performance of this authentication system, we implement an experimental system with Tomcat and AXIS (see Figure 2). In our experimental system, a Web service is developed as the session authority and three Web services are developed to spawn Web service instances that act as session partners in the experiments. The three

services repeatedly invoke each other in sequence until they have spawned a particular amount of services instances and introduced them to the session authority. The instances will be disposed of after they have finished their jobs. This design can prevent the experimental system from consuming large amount of resources and then guarantee the precision of experiment results will not be affected.

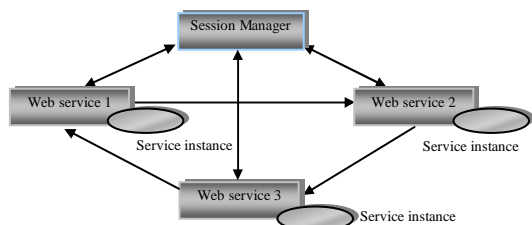


Figure 2. Structure of the experimental System

Figure 3 depicts the results from the experiments in which different multi-party sessions were generated. The numbers of the session partners of these multi-party sessions range from 300 to 1800. As illustrated in Figure 3, the time consumption of invoking and accepting new service instances into a session is proportional to the number of the session partners. This result is consistent with what we obtained in the experiments in [7].

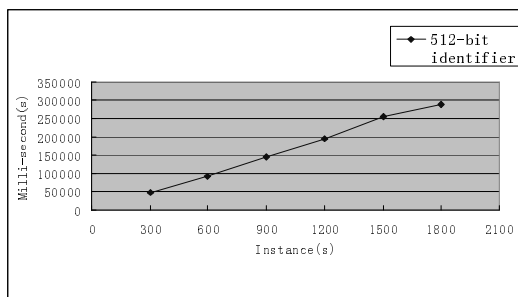


Figure 3. Scalability of the authentication mechanism

In order to examine further the results from the experiment, an analytical model is established. The result from the analytical model also prove time consumption of invoking and accepting new service instances into a session increases linearly as the increase of the number of the session partners. The details of the analytical model is illustrated in [7,8].

Furthermore, we have implemented Hada and Maruyama's scheme proposed in [5] for the purpose of comparison (see Figure 4). This authentication mechanism leverages the Token-based solution to authenticate session partners without prior knowledge of all parties in a multi-party session. In this mechanism, every

session partner shares the same session secret, and so it is able to prevent an instance within a session from communicating with instances in other sessions.

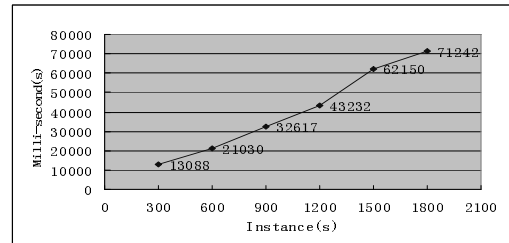


Figure 4. Scalability of the scheme of [5]

As presented in Figure 4, the time consumption of Hada and Maruyama's scheme increases linearly as well. Compared with Hada and Maruyama's solution, the time consumption of our authentication mechanism is relative high. It is because public key algorithms are leveraged in our system. However, the benefits offered by our mechanism are the increased control to the business sessions and better attack confinement.

Our experimental system can intercept and arbitrarily process the messages transmitted among Web services instance. Currently, we are trying to use this experiment system to support our research in addressing the heterogeneity issues of exceptions in Web service business processes.

4. Abstract exception type hierarchy

Web service technology is dedicated to integrate heterogeneous applications through standard protocols. In the service oriented architecture, applications are wrapped with deliberately generated interfaces and users only have limited knowledge about the implementation details of the applications. However, in order to appropriately handle a fault, a forward recovery mechanism must be able to identify the fault, or at least all its consequences [6].

Currently, Web service specifications concentrate on describing the normal functions of Web services and lack enough considerations about the exception handling requirements. In practice, different Web services may be developed with different programming languages, and so the ways of handling exceptions are therefore ad hoc. Corporations have their own rules for defining exception types. The designers of the Web service business processes will find it is difficult to deploy appropriate exception handlers for

business partners especially when the business partners are selected at run time. When an exception is raised in a business session, on many occasions, the session management system cannot understand the semantics of the exception and its consequences.

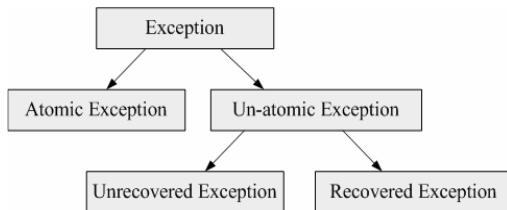


Figure 5. Abstract exception type hierarchy

Figure 5 shows a proposed abstract exception type hierarchy. This hierarchy is able to reflect the consequences of an exception. The root of this hierarchy is *Exception*, which is the super-class of all exception classes. *Exception* has two direct sub-classes: *Atomic Exception* and *Un-atomic Exception*. *Atomic Exceptions* are used to signal that the application keeps atomicity after the raise of the exception, while *Un-atomic Exceptions* indicate that the state of the application has transformed after the exception. The *Un-atomic Exception* class has two sub-classes: *Un-recovered Exception* and *Recovered Exception*. The *Recovered Exception* indicates that the exception has been recovered following certain policies and the application stays in a consistent state. The *Un-recovered Exception* is used to indicate that the system is in an inconsistent state. In our exception handling mechanism, there is an exception handler for every Web service. When the exception message is sent out by a Web service, the associated handler will intercept the message and use the exceptions in the hierarchy to take place the original ones singled by the Web service. Therefore, when designing a business process, the designers can generate the exception handling mechanisms in a standard way.

5. Conclusions

In this paper, we introduce our standardised efforts to address issues in Web service instance identification and exception handling. These issues are critical for generating decentralised Web service business processes. The results from experiments demonstrate that the overhead of execution time introduced by our instance authentication mechanism is proportional to the number of the session partners within a business session. The results derived from an analytical

model further improve the experimental conclusion. Through the development of the experimental system, we have gained a better understanding of instance-level interactions in Web services. Moreover, the experimental system can be extended to support our abstract exception type hierarchy. Current work has generated a solid foundation for our research in the future.

In the future, we are going to 1) keep looking for more efficient algorithms to improve the performance of our authentication system; 2) keep evaluating the practicability of the abstract exception type hierarchy.

References:

- [1] T. Andrews, et al., "Specification: Business Process Execution Language for Web Services Version 1.1," <ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>, 05 May, 2003.
- [2] V. Atluri, S. A. Chun, and P. Mazzoleni, "A Chinese wall security model for decentralized workflow systems," *Proc. the 8th ACM conference on Computer and Communications Security*, pp. 48-57, 2001.
- [3] S. Beryozkin, "Web Services and Sessions," <http://Webservices.xml.com/pub/a/ws/2003/07/22/sessions.html>, July 22, 2003.
- [4] G. B. Chafle, S. Chandra, V. Mann and M. G. Nanda, "Decentralized Orchestration of Composite Web Services," *Proc. the 13th international World Wide Web conference on Alternate track paper & posters*, pp. 134-143, May 2004.
- [5] S. Hada and H. Maruyama, "Session Authentication Protocol for Web Services," *Proc. 2002 Symposium on Application and the Internet*, pp. 158-165, Jan. 2002.
- [6] B. Randell, P.A. Lee, and P.C. Treleaven, "Reliability Issues in Computing Systems Design," *ACM Computing Surveys*, vol. 10, no. 2, pp. 123-165, 1978.
- [7] D. Zhang and J. Xu, "Multi-Party Authentication for Web Services: Protocols, Implementation and Evaluation," *Proc. Symposium on Objected-Oriented Real-Time Distributed Computing*, pp. 227-234, Vienna, 12 May 2004.
- [8] D. Zhang and Jie Xu, "Securing Instance-Level Interactions in Web Services," *Proc. 2005 ISADS IEEE*, pp. 443-450, April 2005.