IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

A Framework for Specifying and Coordinating Reliable Web Services

J.Abinaya[1], A.Appandairaj[2]

[1]PG Student, Department of Computer Science and Engineering, Ganadipathy Tulsi's Jain Engineering College, Kaniyambadi, Vellore-632 102, Tamil Nadu, India

[2]Assistant Professor, Department of Computer Science and Engineering, Ganadipathy Tulsi's Jain Engineering College, Kaniyambadi, vellore-632 102, Tamil Nadu, India

Abstract

Some web services are involved in open environment of the internet, often involve financial, healthcare and other web services that their users require to be trust worthy. The web services business activity (WS-BA) specification standardizes the activation, registration, and coordinator services of web service business activities. A business activity consists of an initiator, a coordinator, and one or more participants. The services are provided by the third-party providers independent of owner. It relies on the use of WS-Security for secure messaging. Moreover, it is possible to integrate WS Reliable Messaging. The user will enter their service id and password for accessing the services. They can view all the services of our application using web services.

Keywords:-*Business activity, Byzantine fault tolerance, distributed transaction service-oriented computing, trustworthy computing, web services*

1. Introduction

Service-Oriented computing and web services are transforming the World Wide Web from a publishing platform into a distributed computing platform, resulting in more and more business activities being conducted online. Such business activities, offered in the open environment of the Internet, often involve financial, healthcare and other web services that their users require to be trustworthy. The web services business activity (WS-BA) specification developed by OASIS, standardizes the activation, registration, and coordinator services of web service business activities. A business activity consists of an initiator, a coordinator, and one or more participants, as shown in, A business activity is started and terminated by the initiator. The initiator propagates the business activity to the participants using a coordination context in the requests.

In this paper, I use a travel reservation application, as a running example, to illustrate the problem we address and our proposed solution. The travel reservation application is a composite web service with which the clients interact directly. Behind the scenes, the application consists of an initiator and two web services, an airline reservation web service and a hotel reservation web service .These two web services are typically provided by third-party service providers, independent of the owner of the travel reservation composite web service.

When a client invokes the composite web service, the initiator creates a web service business activity, in which it communicates with the airline reservation web service to make an airline reservation and the hotel reservation web service to reserve a hotel room. The coordinator component ensures that the business activity is properly propagated to the airline reservation and hotel reservation web services, and is properly terminated according to a predefined policy. The coordinator component provides the activation, registration, and coordinator services. More details on the travel reservation application are provided in, As can be seen from the previous example, the trustworthiness of the coordination service is crucial to the business activities involved. It is very desirable to have the coordinator replicated for Byzantine fault tolerance(BFT). Unfortunately, the WS-BA specification does not specify a standard way for an initiator of a business activity to communicate with the coordinator of the business activity. This design choice makes it easy for vendors to integrate WS-BA coordination functionalities into their business process engines (e.g., the travel reservation application could integrate the initiator and the coordinator into the same process), but makes it difficult to ensure interoperability among the initiators and coordinators of different vendors. Such interoperability is needed because of the benefits of separating the initiator and the coordinator in particular. A small company might create a composite web service (such as a travel reservation service) for its customers. However, it might be difficult for them to implement and host robust coordination services. Even if some of them do offer coordination services, other web services providers.

2. System description and Methodology

I present the lightweight BFT algorithm and associated mechanisms for achieving trustworthy coordination of WS-BA. An important objective is to enable an independent third party to launch trustworthy coordination services for WS-BA that span multiple enterprises. The practicality of the BFT solution is essential for real-world use, which means that it must be lightweight with moderate runtime overhead and good scalability.

Traditional BFT state-machine replication mechanisms can be used to mitigate the threats to the WS-BA coordination services. However, it is not wise to use them naively. Such mechanisms are designed to protect general-purpose state full servers from Byzantine faults. They impose a total order on incoming requests and, thus, are very heavy weight and incur significant runtime overhead, due to the multiple rounds of message exchange that are required. In our lightweight BFT algorithm, we exploit knowledge of the state model of the WS-BA coordination services, described below, and use a solution that is customized to this specific application. In short, the lightweight BFT algorithm requires only source ordering, instead of total ordering, of incoming requests, which eliminates inter-replica message exchange and hence significantly reduces the communication steps and processing overhead, compared with traditional BFT algorithms.

2.1 Methodology

2.1.1Trustworthy Implementation

Cloud Services and information services will only be truly pervasive when they are so dependable that we can just forget about them. In other words, at a time where cloud storage is starting to find their way into just about every aspect of our life, we need to be able to trust them. Yet the way we build cloud storage, and the way that we now build services around those cloud storage. The Goals consider trust from the user's point of view. The key questions are: Is the technology there when I need it? Does it keep my confidential information safe? Does it do what it's supposed to do? And do the people who own and operate the business that provides it always do the right thing?

These are the goals that any Trustworthy Computing has to meet:

Goals	**The basis for a customer's decision to trust a system**
Security	The customer can expect that systems are resilient to attack, and that the confidentiality, integrity, and availability of the system and its data are protected.
Privacy	The customer is able to control data about themselves, and those using such data adhere to fair information principles.
Reliability	The customer can depend on the product to full fill its functions when required to do so.
Business Integrity	The vendor of a product behaves in a responsive and responsible manner.

- **Security**: An intruder cannot render my system unusable or make unauthorized alterations to my data.
- **Privacy**: My personal information isn't disclosed in unauthorized ways. When I provide personal information to others, I am clearly informed of what will—and won't—be done with it, and I can be sure they will do what they promise.
- **Reliability**: When I upload new data, I don't have to worry about whether it should be easily accessible with my existing environment. I can read and forward my data as email whenever I want by clicking the trusted link. I never get "system unavailable" messages.
- **Business Integrity**: My service provider responds rapidly and effectively when I report a problem.

2.1.2 Light Weight Byzantine Fault Tolerance

Light weight Byzantine fault tolerance is a sub-field of fault tolerance research inspired by the **Byzantine Generals' Problem**, which is a generalized version of the Two Generals' Problem.

The objective of light weight Byzantine fault tolerance is to be able to defend against *Byzantine failures*, in which components of a system fail in arbitrary ways (i.e., not just by stopping or crashing but by processing requests incorrectly, corrupting their local state, and/or producing incorrect or inconsistent outputs.). Correctly functioning components of a Byzantine fault tolerant system will be able to correctly provide the system's service assuming there are not too many Byzantine faulty components.

A Byzantine fault is an arbitrary fault that occurs during the execution of an algorithm by a distributed system. It encompasses both *omission failures* (e.g., crash failures, failing to receive a request, or failing to send a response) and *commission failures* (e.g., processing a request incorrectly, corrupting local state, and/or sending an incorrect or inconsistent response to a request). When a Byzantine failure has occurred, the system may respond in any unpredictable way, unless it is designed to have Byzantine fault tolerance.

For example, if the output of one function is the input of another, then small round-off errors in the first function can produce much larger errors in the second. If the second function were fed into a third, the problem could grow even larger, until the values produced are worthless. Another example is in compiling source code. One minor syntactical error early on in the code can produce large numbers of perceived errors later, as the parser of the compiler gets out-of-phase with the lexical and syntactic information in the source program. Such failures have brought down major Internet services. For example, in 2008 Amazon S3 was brought down for several hours when a single-bit hardware error propagated through the system.

In a Byzantine fault tolerant (BFT) algorithm, steps are taken by processes, the logical abstractions that represent the execution path of the algorithms. A faulty process is one that at some point exhibits any of the above failures. A process that is not faulty is correct.

The Byzantine failure assumption models real-world environments in which computers and networks may behave in unexpected ways due to hardware failures, network congestion and disconnection, as well as malicious attacks. Byzantine failure-tolerant algorithms must cope with such failures and still satisfy the specifications of the problems they are designed to solve. Such algorithms are commonly characterized by their resilience t, the number of faulty processes with which an algorithm can cope.

Many classic agreement problems, such as the Byzantine Generals' Problem, have no solution unless $n \geq 3t + 1$, where n is the number of processes in the system. In other words, the algorithm can ensure correct operation only if fewer than one third of the processes are faulty.

2.1.3 Travel Reservation Example

Fig 1 shows the normal execution steps of the travel reservation application using the WS-BA specification and the WS-BA-I extension.

In this example, I use the atomic-outcome coordination type and the BAwCC protocol. The travel reservation application includes the airline reservation and hotel reservation web services. The airline reservation web service comprises an airline service and a participant service, and the hotel reservation service comprises a hotel service and a participant service.

The initiator creates a business activity coordination context using the activation service (1 and 2), and then registers with the registration service (3 and 4). Next, the initiator searches the airline reservation web service for an offer, from which it receives a reply (5 and 6), and then it searches the hotel reservation web service for an offer, from which it receives a reply (7 and 8). Next, it invokes the coordinator service to create tickets for the participants (9and 10).

Using the invitation tickets, it books an airline reservation (11), and a hotel room. On receiving a reservation request from the initiator, the airline reservation web service registers with the registration service(12); similarly, the hotel reservation web service registers with the registration service.

The initiator sends a Complete Participants message to the coordinator service. The coordinator sends an acknowledgment to the initiator as soon as it has sent a complete message to the participant services of the airline reservation and hotel reservation web services (13) and without waiting to receive their completed messages (16).

As such, to know the participants' state, the initiator must query the coordinator service. When all of the participants are in the correct state, the initiator sends a Close Participants message (17) to the coordinator service. Similar to the handling of the Complete Participant request, the coordinator sends an acknowledgment immediately after it has transmitted a close message to the participant services of the airline reservation and hotel reservation web services(18),without waiting to receive their completed messages.

If the coordination services are not highly available, the airline reservation and hotel reservation web services might not be able to register for the WS-BA. Moreover, the coordination services might not be able to complete and close the WS-BA .If the integrity of the coordination services are Compromised, the WS-BA might have inconsistent outcomes(and other undesirable results) for the airline reservation and hotel reservation web services.

Sequence Diagram for Travel Reservation

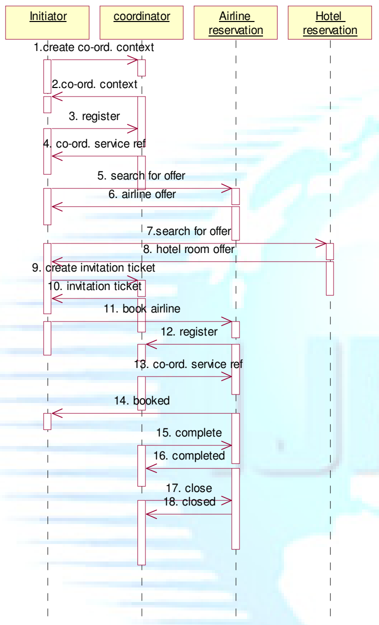
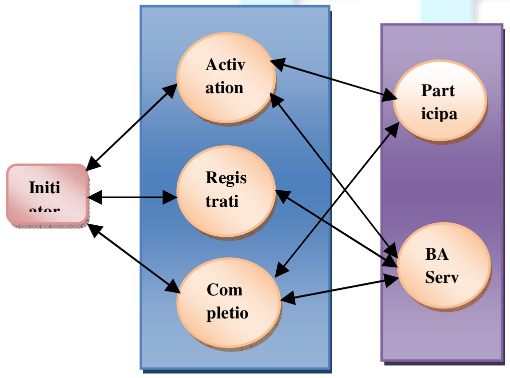


Fig1.Travel Reservation Example

3. System Design

3.1 Architectural Framework



3.1.1 Initiator & Activator

The initiator searches the cloud storage web service for an offer, from which it receives a reply, and then it searches the data integrity web service from which it receives a reply. Next, it invokes the coordinator service to create token for the participants. Using the invitation token, it uploads a data and downloads data. On receiving a token request from the initiator, the cloud storage web service registers with the registration service; similarly the data integrity web service registers with the registration service. The initiator sends a Complete Participants message to the coordinator service. The coordinator sends an acknowledgment to the initiator as soon as it has sent a complete message to the participant services of the cloud storage and data integrity web services without waiting to receive their completed messages. The Activation service creates a Coordinator object and a cloud storage context for all cloud storage. Essentially, the Activation service haves like a factory object that creates Coordinator objects.

3.1.2 Coordinator

The Coordinator service runs the 2PC protocol, which ensures atomic commitment of the distributed cloud storage. The coordination service providers uses lightweight BFT algorithm because of its relatively low overhead. For the best fault isolation, it is desirable to deploy the coordinator replicas across geographically separated data centres, which are readily available for the large companies. When cloud storage is activated, a Coordinator object is created. The Coordinator object provides the Registration service, the Completion service and the Coordinator service. The cloud storage context contains a unique cloud storage id and an endpoint reference for the Registration service, and is included

3.2.1 Bft Coordination Services

Traditional BFT state-machine replication mechanisms can be used to mitigate the threats to the WS-BA coordination services. However, it is not wise to use them naively. Such mechanisms are designed to protect general-purpose stateful servers from Byzantine faults. They impose a total order on incoming requests and, thus, are very heavyweight and incur significant runtime overhead, due to the multiple rounds of message exchange that are required. State model of the WS-BA coordination services, described below, and use a solution that is customized to this specific application. In short, the lightweight BFT algorithm requires only source ordering, instead of total ordering, of incoming requests, which eliminates inter replica message exchange and hence significantly reduces the communication steps and processing overhead, compared with traditional BFT algorithms.

A participant might encounter a problem or fail during the processing of the business activity. If an error occurred when the participant was trying to complete its normal activity, it generates a fail message and sends that message to the coordinator, possibly on recovery from a transient fault. Similarly, if an error occurred when the participant was trying to cancel or compensate an activity, it generates a cancel complete message and sends that message to the coordinator.

If the coordination protocol supports it then the coordinator will execute a particular coordination protocol(specified by a protocol URI)on the currently enlisted participants, upon receiving the coordinate message at any time prior to the termination of the coordination scope. This message instructs the Activity Coordinator to send protocol messages to all of the registered Participants; since the coordinator may be invoked multiple times during the lifetime of an activity, it is possible that different protocol messages may be sent each time coordinate is called. Once the Participants have processed the messages and returned outcomes, it is up to the Activity Coordinator to consolidate these individual outcomes into a single result, which is sent to the Client Respondant via the coordinated message. If there is no Activity associated with the context then the invalid Coordinator message will be generated. Because this operation can be used to cause messages to be sent to Participants at times other than when the Activity completes, the implementation of the coordinator must ensure that such messages clearly identify that the Activity is not completing. If the Activity has begun completion, or has completed, then the invalid Activity message is sent to the Client Respondant.

4. CONCLUSION

I have presented a comprehensive study of the threats to the coordination services of WS-BA. I have carefully analysed the state model of the WS-BA coordination services, and have argued that it is unnecessary to perform total ordering of requests at the coordinator replicas, to achieve trustworthy coordination of WS-BA. Rather, it suffices to ensure that messages are delivered in source order, i.e., the order in which the sender sent them. This analysis has enabled us to develop a lightweight BFT algorithm that mitigates the threats and avoids the runtime overhead associated with traditional BFT algorithms. I have implemented the lightweight BFT algorithm and associated mechanisms, and have incorporated them into the open-source Kandula framework that implements the WS-BA specification and the WS-BA-I extension. The performance evaluation results obtained using the prototype implementation show moderate overhead, especially in a wide-area network, where WS-BA are typically deployed.

References

[1]D.Davis, A.Karmarkar, G.Pilz, S.Winkler,andU.Yalcinalp,WebServices Reliable Messaging(WS-Reliable Messaging)Version1.1,OASIS Standard, Jan. 2008.

[2] D. Dolev, "The Byzantine Generals Strike Again," J. Algorithms,vol. 3, no. 1, pp. 14-30, 1982.

[3] D. Dolev, "An Efficient Algorithm for Byzantine Agreement without Authentication," Information and Control, vol. 52, no. 3, pp. 257-274, Mar. 1982.

[4] D. Dolev and H.R. Strong, "Authenticated Algorithms for
Byzantine Agreement," SIAM J. Computing, vol. 12, pp. 656-666, 1983.

[5]W.Zhao and H. Zhang, "Byzantine Fault Tolerant Coordination for Web Services Business Activities," Proc. IEEE Int'l Conf.Services Computing, pp. 407-414, July 2008.

[6] M. Feingold and R. Jeyaraman, Web Services Coordination,Version 1.1, OASIS Standard, July 2007.

[7] T. Freund and M. Little, Web Services Business Activity, Version 1.1,OASIS Standard, Apr. 2007.

[8]H.Garcia-Molina, F.Pittelli, and S.Davidson,"ApplicationsofByzantine Agreement in Database Systems," ACM Trans.DatabaseSystems, vol. 11, no. 1, pp. 27-47, 1986.

[9] Y. Gil and D. Artz, "A Survey of Trust in Computer Science and The Semantic Web," J. Web Semantics, vol. 5, pp. 58-71, 2007.

[10] T. Grandison and M. Sloman, "A Survey of Trust in Internet Applications," IEEE Comm. Survey Tutorials, vol. 4, no. 4, pp. 2-16,
Oct.-Dec. 2000.