Chin-Wan Chung Chong-Kwon Kim
Won Kim Tok-Wang Ling
Kwan-Ho Song (Eds.)

Web and Communication Technologies and Internet-Related Social Issues – HSI 2003

LNCS 2713

Second International Conference on Human.Society@Internet
Seoul, Korea, June 2003
Proceedings

Human.Society | Internet

Springer

Misc Topics in Communications

Short Contributions

The Framework for the Message Transport Agent in a Web-Based Information System

Kyung-Chang Kim[1], Young Chul Kim[2], and Hewon Lee[3]

[1] Dept. of Computer Engineering, Hongik Univ.
kckim@cs.hongik.ac.kr
[2] Dept. of Computer & Info Comm., Hongik Univ.
bob@wow.hongik.ac.kr
[3] Info. Support Division, ETRI
hewlee@etri.re.kr

Abstract. The main role of the messaging system or message transport agent (MTA) in a Web-based information system is the correct and reliable processing of applications serviced by the system in the presence of system failures. In this paper, we define a general framework for the MTA in a Web-based information systems based on the XML platform. Two important issues that need to be addressed by the MTA are (1) the message protocol to be used to ensure interoperability and (2) the technique used to ensure database consistency in processing user applications. The framework for the MTA is defined by how the two issues are addressed and we show the design of the MTA. A prototype system has been implemented based on the design.

1 Introduction

The messaging system or MTA in a Web-based information system [3] plays a very important role since it is responsible for the correct and reliable execution of Web-based services. Two important issues related to the MTA are the message protocol to be used by the MTA for message exchange to ensure interoperability and the technique to be used to enable correct and reliable processing of Web service applications in the presence of system failures.

In this paper, we define a general framework for the MTA in a Web-based information system. In the framework, we present the message protocol for the MTA and then propose a technique to be used by the MTA for correct processing of user requests that ensure database consistency. We claim that the framework defined in the paper has not been published elsewhere and hence no related works have been mentioned.

The organization of the paper is as follows. In Chapter 2, we describe the architecture of a typical Web-based information system that we will adopt throughout the paper. Chapter 3 defines the framework for the MTA and the design of the MTA

C.-W. Chung et al. (Eds.): HSI 2003, LNCS 2713, pp. 582-587, 2003.

based on the framework. Chapter 4 describes the implementation environment for the MTA including the message-passing paradigm and a summary is given in Chapter 5.

2 Architecture of a Web-Based Information System

Fig. 1 shows the overall architecture of an e-logistics system, the web-based information system that we will adopt in our paper. Without loss of generality, the e-logistics system is a typical Web-based distributed information system. A *client* of the e-logistics system can be a local post office, customer or client, a transport company or a related company. A request from the client (web browser) is sent to the MTA (web server) through the Work Flow Management Server (WFMS). The MTA analyzes each request to determine what tasks need to be carried out by the various database servers in the e-logistics system, and then creates sub-requests to be sent to the appropriate servers for processing.

There are several database servers in the e-logistics system. One is the Track and Trace Server (TTS) and another is the Moving Object Management Server (MOMS). The request for tracking information, such as the current location of a specific package on delivery is processed by the TTS. The MOMS processes request for information on some specific transport vehicle. There may be other database servers in the e-logistics system but are skipped in this paper.

The detailed message flow involved in servicing a client request by the MTA in the e-logistics system is described as follows.

1. A user request is initiated from a Web browser and sent to the WFMS. The request may be in a HTML form.
2. The WFMS transforms the request into an XML message form based on Simple Object Access Protocol (SOAP). This SOAP request is then sent to MTA.
3. The MTA analyzes the SOAP request to determine which servers are involved in the request and send SOAP requests to the corresponding servers.
4. Upon receiving a SOAP request, the TTS or MOMS initiates a database query and accesses the local database through OLEDB interface. The result (i.e. return value) is returned to the MTA through a SOAP response.
5. The MTA checks all return values from MTA and MOMS and send a *commit* message back to the corresponding servers as a SOAP request if there are no errors. Otherwise, the MTA send an *abort* message as a SOAP request.
6. Upon receiving a SOAP request, the TTS or MOMS initiates a database commit or abort action depending on whether it received a commit or an abort message and then sends an *Ack* message back to the MTA as a SOAP response.
7. If the return values have no errors, the result is sent to WFMS as a SOAP response.
8. The result is transformed into a HTML form and sent back to the initial Web browser for presentation.

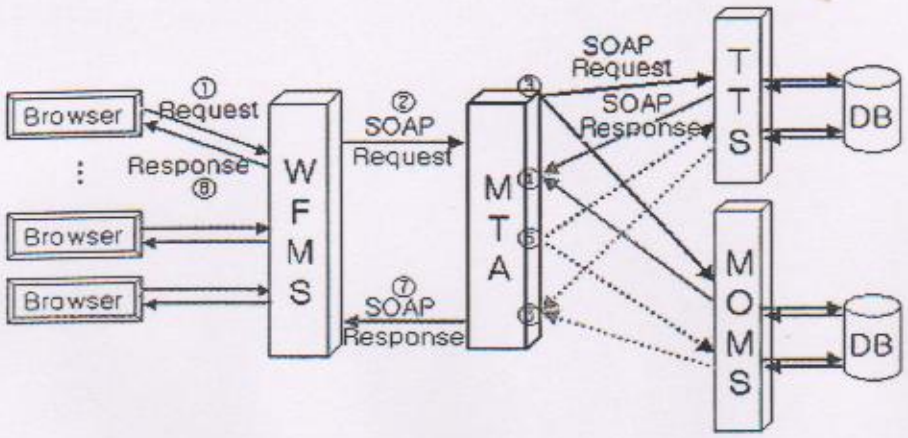

Fig. 1. Architecture of an e-logistics system with message flow

3 Designing the MTA

The two issues that need to be addressed in the design of the MTA are one, the message protocol to be adopted by MTA for interoperability and two, the technique used to ensure database consistency during the servicing of a user request by the participant servers in the presence of system crashes and failures.

3.1 Message Protocol

The ebXML Message Service specification [1] is the first, truly open, standards track specification that has been designed to be suitable for use by businesses for the purpose of the secure, reliable exchange of any manner of electronic business information with their partners, suppliers and customers. Effective with the version 0.98 draft of the ebXML Message Service specification, published in March of 2001, the ebXML Message Service is defined as a set of layered extensions on the SOAP1.1 [4]. Thus, the ebXML Message Service provides the necessary extensions for security and reliability that are not addressed directly by the SOAP1.1 specification. In this context, the basic approach of SOAP and ebXML Message Service are the same.

In this paper, we adopt SOAP, a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an *envelope* that defines a framework for describing what is in a message and how to process it, a set of *encoding rules* for expressing instances of application-defined data types, and a *convention* for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in [4] describe how to use SOAP in combination with HTTP and HTTP Extension Framework. The advantages of SOAP are that the protocol is platform independent, language neutral, firewall friendly, lightweight, strictly follows the W3C specification and easy to implement.

3.2 Technique for Ensuring Database Consistency

When a database request is made in servicing a user request, the local database may be in an inconsistent state due to a system crash or a network failure. We propose a variation of the *distributed two-phase commit protocol* in the message exchange model to solve the problem of database inconsistency when a user request is serviced by the MTA.

In the traditional distributed two-phase commit (2PC) protocol [2,5], the *coordinator* sends a "prepare" message to all *participants* in order to decide whether to commit or abort the distributed transaction. Upon receiving the "prepare" message, each participant sends either a "Yes" or a "No" reply message to the coordinator based on whether it can either commit or abort its local transaction. This is the *first phase* of the commit protocol. Based on the result of reply messages from all the participants, the coordinator decides whether to commit or abort the transaction. If it receives a "No" reply message from any of the participants, it aborts the transaction. Otherwise, it commits the transaction. If the coordinator decides to commit/abort the transaction, it sends a "commit"/"abort" message to all participants. Upon receiving a "commit"/"abort" message, each participant acts accordingly and sends an "ACK" reply message to the coordinator. This is the *second phase* of the commit protocol.

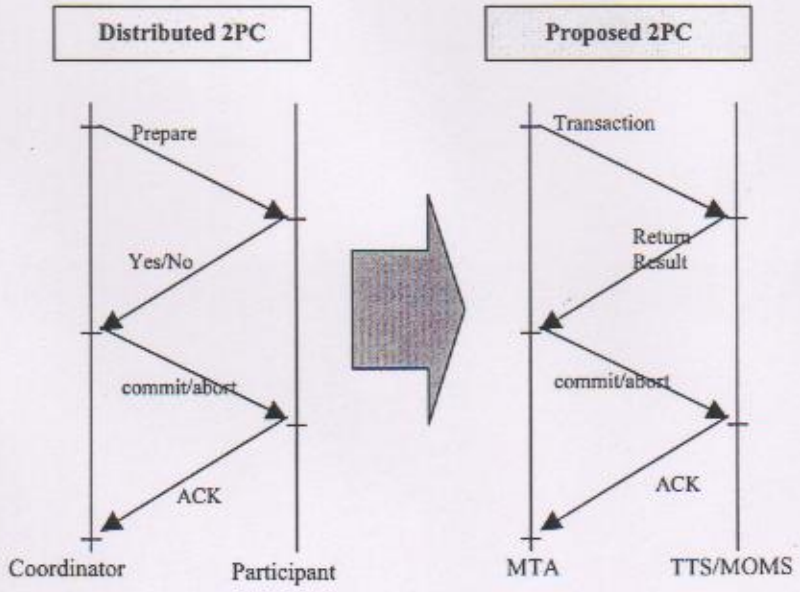

Fig. 2. Proposed distributed 2PC Protocol

In this paper, we modify the traditional distributed 2PC protocol to coordinate whether to commit or abort the client request serviced by MTA. In the *first phase*, a sub-request (i.e. transaction) of the client request is sent in a message form from the MTA to each participant server and each participant returns the result of the sub-request as a reply message. The MTA then decides whether to commit or abort the

user request based on the return results. In the *second phase*, the MTA sends a "commit"/"abort" message to each participant server. Upon receiving a "commit"/"abort" message, each participant acts accordingly and sends an "ACK" reply message to MTA. Fig. 2 shows the variation of the distributed 2PC protocol just described. We claim that it is easy to adopt the proposed 2PC protocol to be used by the MTA of any other Web-based distributed information system.

4 Implementation Environment for the MTA

Due to the stateless nature of the web environment, there is some difficulty in implementing the message-passing paradigm between the MTA and either the TTS or the MOMS. The technique we use is to divide the web pages by phases in executing under the proposed 2PC protocol such that there is a web page to determine whether it is possible to service the sub-request and another web page to do the actual servicing of the sub-request.

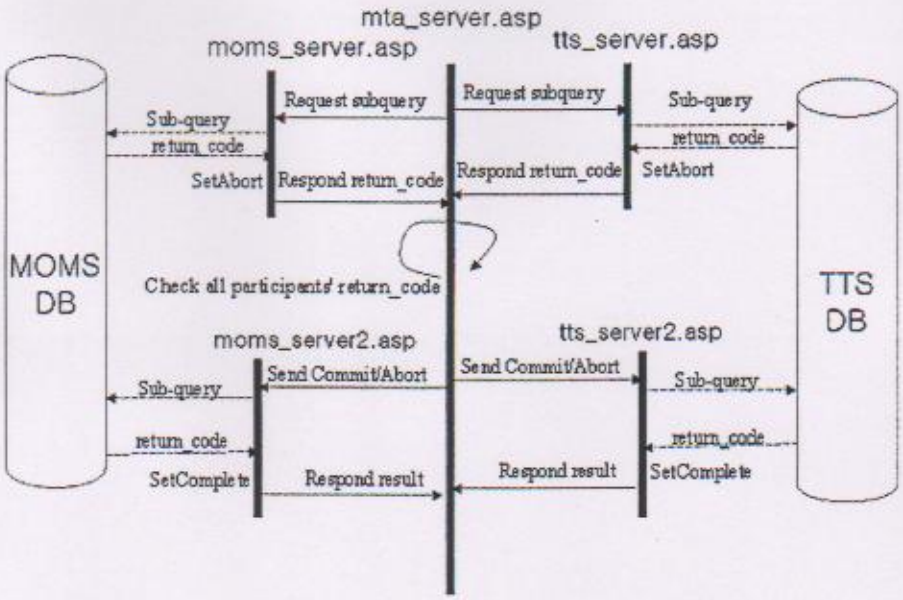

Fig. 3. Implementation of Message Paradigm between MTA and TTS/MOMS

Fig. 3 shows the implementation of the message-passing paradigm between the MTA and TTS/MOMS. The routine mta_server.asp resides in the MTA server while the routines tts_server.asp and tts_server2.asp and the TTS DB reside in the TTS server. To deliver the return code to MTA, the TTS server calls tts_server.asp. The tts_server.asp executes the sub-query and sends its return code, i.e. success or failure of the sub-query, to MTA after a request to TTS DB has been completed while aborting its local transaction (or sub-request) to suspend the execution of the

transaction momentarily. The message-passing paradigm between MTA and MOMS is identical to the one between MTA and TTS described provided that the routines moms_server.asp, moms_server2.asp and MOMS DB reside in MOMS server.

After receiving the return code from TTS/MOMS, MTA decides whether to commit or abort the client request. To send its decision, i.e. whether to commit or abort, to TTS/MOMS it calls tts_server2.asp/moms_server2.asp. The execution of tts_server2.asp/moms_server2.asp results in re-executing the sub-query in TTS/MOMS. The return result sent back to MTA as a reply message is analyzed by MTA to form the final result to send back to WFMS.

In summary, the message exchange between MTA (the coordinator) and TTS/MOMS (the participant servers) does not occur continuously between the web page in MTA and the initial web page executed in TTS/MOMS. The web pages executed in TTS/MOMS are divided into a web page to determine the success or failure of a sub-request and another web page to re-execute the sub-query in case MTA has decided to commit the client request. We implemented a prototype system based on the concepts described above.

5 Summary

A Web-based information system such as an e-logistics system must provide a high quality customer service in the management and processing of data in a Web environment. One important task of the MTA for such a system is to provide an efficient message exchange service in processing a client request correctly and reliably. The framework for the MTA of a web-based information system involves the message service protocol to be used to provide interoperability and the technique used to ensure database consistency in servicing a client request in the presence of system crashes and network failures.

In this paper, we first defined the framework for the MTA of a Web-based information system by adopting SOAP as the message service standard protocol. In addition, we proposed a variation of the distributed 2PC protocol to be used to ensure database consistency when a client request, which requires a database service, is processed in the presence of network failures and system crashes. The design and implementation of the MTA for the e-logistics system based on the defined framework is given.

References

[1] Ferris, Christopher: ebXML Message Service. http://www.gca.org/papers/xmleurope2001/papers/html/s09-3.html
[2] Gupta, Ramesh, et al.:Revisiting Commit Processing in Distributed Database Systems. Proc. ACM SIGMOD Intl. Conf. (1997) 486-497
[3] Lee, Hewon, et al.: e-Logistics Integrated Platform MTA Requirement Specification. ETRI Internal Document, Oct. (2001)
[4] Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/SOAP
[5] Ramakrishnan, R., Gehrke, J.:Database Management Systems. 2nd edn. McGraw-Hill (2000)