Reliability and Message Security for Distributed Web Service Handlers

Beytullah Yildiz
Department of Computer Engineering
TOBB Economics and Technology University
Ankara, Turkey
E-mail: byildiz@etu.edu.tr

***Abstract*—Web Service handlers are supportive functionalities and capabilities to the service endpoint such as security, reliability and logging. In the common usage, they perform their executions in a single memory space with the service endpoint. However, by a suitable structure, they can be distributed to increase availability, scalability and performance. On the other hand, the distribution necessitates additional mechanisms to provide essential service quality. In this paper, reliability and message security for the distributed Web Service handler will be investigated. The benchmark results are provided to illustrate that the utilized reliability and security mechanisms of the messaging for the handler distribution are reasonable. With the fair cost, Web Service handlers are reliably and securely distributed.**

Keywords-Web Service; distributed computing; replication; reliability; security

I. INTRODUCTION

Web Service is a technology providing seamless and loosely coupled interactions, which help to build platform independent distributed systems. Web Service is considered to be an ideal technology to provide new IT architectures. It is claimed that the age of proprietary information systems has come to an end and the age of shared services is already on its way [1]. In this new era, companies obtain or outsource their IT capabilities in order to reduce the cost, deploy solutions faster, and create new opportunities.

Software standards and communication protocols providing the common languages are at the foundation of Web Service. Information is easily exchanged between different applications via these standards and protocols. In short, Web Service provides opportunities so that diverse and distributed applications can communicate with each other in a standard way.

Web Service integrates an endpoint and handlers in a common framework. It employs supportive functionalities and capabilities, called as Web Service handlers, to provide a full-fledged service. These capabilities might be related to security, reliability, orchestration, logging as well as any necessary capabilities for a distributed system. A Web Service may employ several handlers in a single interaction. In other words, a chain of handlers can contribute to a service execution. With these additive functionalities, Web Services aim to offer better environment. On the other hand, overloading a service with required supportive functionalities, inevitable for many cases, may cause degradation in the service quality. A service endpoint with many handlers may suffocate in a single memory space. Hence, it is wise to use additional computing power. This brings the idea of distribution. There are different reasonable approaches for the Web Service handlers for the distribution. Some suggest that they can be distributed as services; others create a specific distributed environment for them. By creating a specific environment, a distributed handler operating system provides a better environment, especially, when the concern is performance. However, the distribution requires additional mechanisms to provide the suitable environment.

Security and reliability are among the most important criteria that need to be considered when a distributed system is being evaluated. This paper investigates reliability and message security for the distributed Web Service handlers and their effect over the system performance. The rest of this paper is organized as follows. Section II provides information about the related works of reliability and security. Distributed Web Service handler execution is briefly explained in Section III. Section IV investigates reliability. Section V gives details about the message security. Finally, the paper will be concluded in Section VI.

II. RELATED WORKS

Reliability and security are very important for the distributed applications. Many researches on security and reliability have been conducted for the distributed applications in [2] [3] [4] [5].

For Web Services, several standards are provided for the security and reliability purpose: WS-Security [6], WS-ReliableMessaging [7]. WS-Security addresses security by leveraging existing standards and provides a framework to imbue these mechanisms into a SOAP message. This happens in a transport-neutral fashion. WS-Security defines a SOAP header element to carry security related data. This header element contains the information defined by XML signature that conveys how the message was signed, the key that was used, and the resulting signature value. Likewise, the encryption information can be inserted to the SOAP header. In short, WS-Security presents an end-to-end solution for Web Service security by keeping all security information in the related SOAP header element.

 ISBN: 978-1-61208-200-4

The WS-ReliableMessaging specification offers an outline to ensure reliable message delivery between the sender and receiver. The specification provides an acknowledgement based scheme to guarantee that data are transferred between the communicating entities. Although it is for the point-to-point communication, the specification also supports service composition and transactional interaction.

III. DISTRIBUTION

Web Service handlers are executable in the distributed environment to meet necessary requirements and to provide enough computing power for Web Services. Distribution improves scalability, availability and performance of the overall system. On the other hand, it brings challenges. A manager for the distributed Web Service handlers must be employed to organize the execution which contains an orchestration mechanism, explained in [8]. The manager also requires a decent execution engine to meet the performance requirements. The details of the manager are provided in [9]. The distribution overhead must be acceptable, which is investigated in [10].

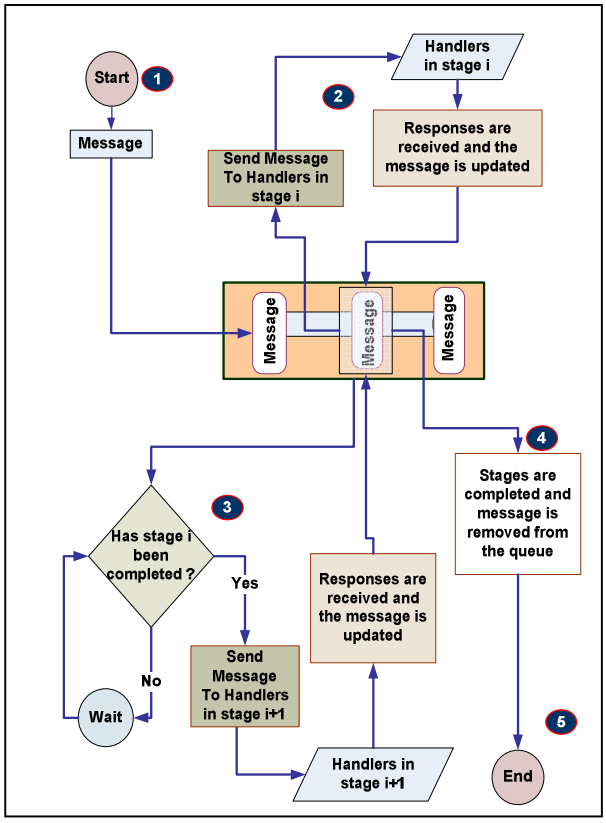

Figure 1. Executing the messages in the distributed Web Service handlers.

The execution of the messages is shown in the Figure 1. Messages, stored in a processing queue, are executed concurrently. Manager ensures that each message is executed without being interrupted by the remaining messages in the queue. Every message execution contains stages, which host the distributed handlers.

A message in the processing queue is instantly sent to all handlers of a stage. The handlers in a stage are executed in a parallel manner. The manager waits the completion of the handler executions before starting the delivery of the message to the next stage. This procedure continues until all stages of a message are completed. In this process, since handlers are deployed to the remote machines, the security and reliability of the messaging become important. The reliability of a handler itself is also essential for the successful execution.

IV. RELIABILITY

Software reliability is described as the probability that the software functions without failures under given conditions during a specified period of time [11]. Reliability is also measured in terms of percentage of failure circumstances in a given number of attempts to compensate for variations in usage over time [12]. For Web Services, although reliability is viewed by some researchers as a non-functional characteristic [13], Zhang and Zhang describes one of the more comprehensive definitions of Web Services reliability, which is defined as a combination of correctness, fault tolerance, availability, performance, and interoperability, where both functional and non-functional components are considered [14].

In this paper, the reliability will be investigated in two sections: the reliability originating from the handler replication and the reliability coming from the utilization of a reliable messaging system.

A. *Replicating handlers*

Replication is critical to reliability, mobility, availability, and performance of a computing system. We benefit from the replication in our daily life too. Even our body benefits from the replications; we have two legs, hands, eyes and ears. We keep a spare tire in our car to replace a flat one in an emergency. The important files are backed up to reduce the probability of lost. Software systems also utilize the same strategy by replicating the data and the computing nodes.

There are basically three replications: data, process and message. These concepts are extensively explored in [15]. Data replication is the most heavily investigated one. However, the other replications are also very important in the distributed systems, especially for Service Oriented Architecture.

The process replication is particularly main interest in this paper because the intention is to investigate the replication of the handlers. There exist two main approaches in this area. The first one is *modular redundancy* [16]. The

 ISBN: 978-1-61208-200-4

second approach is called *primary/standby* [17]. *Modular redundancy* has the replicated components that perform the same functionalities. All the replicas are active. On the other hand, *primary/standby* approach utilizes a primary replica to perform the execution. The remaining replicas wait in their standby state. They become active when the primary replica fails.

The processes can be classified in two categories; no consistency and consistency. The fist category is the simplest one; the processes are stateless. They do not keep any information for the processed data. Therefore, the consistency is not an issue between the processes. Replicated instances can be allowed running concurrently. On the other hand, replicas may enter in an inconsistent state if the process is not atomic and statefull. Inconsistency have been extensively investigated in [18].

Replication is a very important capability where a handler is inadequate. Sometimes, a handler may not be sufficient to answer the incoming requests. The tasks may line up so that the overall performance degrades. This is similar to a shopping center where the customers are waiting in the line to be served. The solution is to add one more person to serve when it is necessary. Similarly, adding a handler to help the execution contributes the overall performance.

In addition to the performance, a replica can be leveraged for fault tolerance. It is possible that a handler crashes. The replication contributes to the continuity of the execution and improves availability and reliability of the service. Without using handler replication in the case of an error, the whole computation cannot continue. The computation becomes more resilient with the handler replication. The execution continues while at least one replica of every handler has not failed.

For N handlers with the replication factor of R, the execution can be successful for R-1 failures per handler. The maximum allowable number of error is:

$$\sum_{i=1}^{N} R_i - 1 \qquad (1)$$

where N is the number of handlers, R_i is the replication number of ith handler. The system cannot continue its execution even in a single handler fault where $\forall i \in N: R_i = 1$.

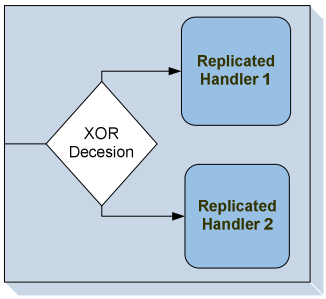

Figure 2. Replicated handler execution; only one of the handlers can be executed.

In the distributed Web Service handler execution environment, a variation of *primary/standby* approach is utilized. The replicas are prioritized. The handler having highest priority is assigned to execute a message. The other replicas wait until their priorities become highest. The system is able to change the priority during the execution. When a fault occurs, the handler priority is minimized. The replicas are never allowed to be executed concurrently unless they are the instance of the stateless handlers. Even though they are allowed to run in parallel manner, they cannot join the processing of the same message. The messages have to be different so that the parallel execution does not cause inconsistency.

When only one of the several replicated handlers is executed, shown in Figure 2, the following formula works for the reliability:

$$R_{RH} = \sum_{i=1}^{n} P_i \, R_i \qquad (2)$$

where R_{RH} is the reliability of the replicated handlers' execution, P_i is the execution probability of the handler i and $\sum_{i=1}^{n} P_i = 1$

The reliability of parallel handlers with AND junction and the reliability of serial handlers can be formulated as:

$$R_s = \prod_{i=1}^{n} R_i \qquad (3)$$

where $R_s$is the reliability of the handlers' execution and R_i is the reliability of the handler i.

By using Formulas 2 and 3, the reliability of handlers' execution in Figure 3 can be formulated as:

$$R_s = \prod_{i=1}^{2} R_i * \sum_{Ri=1}^{2} P_{Ri} R_{Ri} \; * R_3 \qquad (4)$$

$$R_s = \prod_{i=1}^{3} R_i * \sum_{Ri=1}^{2} P_{Ri} \, R_{Ri} \qquad (5)$$

where R_s is the reliability of handlers' execution. Ri is the ith replica and $P_{Ri} = 1$ for only one replicated handler, which is executed, and the value is 0 for the remainders.

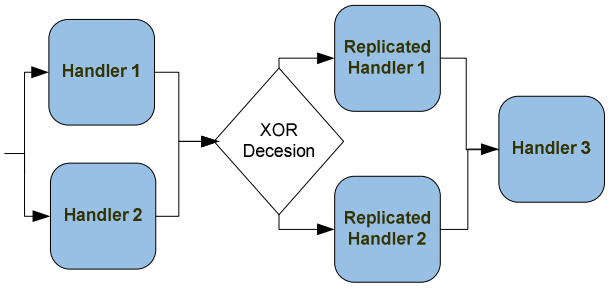

Figure 3. A sample configuration for the handlers' excution

B. Reliable messaging

The distributed handler mechanism benefits from two different sources for the reliability of the message delivery: a messaging broker, and its own mechanism.

The messaging system, NaradaBrokering, provides a message level reliability. It also offers supportive functionalities for the messaging and grants very reasonable performance [19]. The messages can be queued up to several thousands and are gradually delivered to their destinations to provide a flow control for the messaging. Additionally, it has Reliable Delivery Service (RDS) component that delivers payload even if a node fails [20].

RDS stores all the published events that match up with any one of its managed templates, which contain the set of headers and content descriptors. This archival operation is the initiator for any error correction, which is caused by the events being lost in transit to their targeted destinations and also by the entities recovering either from disconnect or a failure. For every managed template, RDS also maintains a list of entities for which it facilitates reliable delivery. RDS may also manage information regarding access controls, authorizations and credentials of the entities that generate or consume events, which are targeted to this managed template.

When an entity is ready to start publishing events on a given template, it issues a discovery request to find out the availability of RDS that provides archival environment for the generated template events. The publisher will not circulate template events until such time that it receives a confirmation that RDS is available.

The publisher ensures that the events are stored by RDS for every template event that it produces. After successful delivery of the event to RDS, it is archived and a message is sent to the publisher to verify that the message is received by RDS successfully. Otherwise, a message of failure with the related event id is sent back to the publisher. After having the verification, the suitable matching engine is utilized to compute the destinations associated with the template event.

A subscriber registers with RDS. A sequence number linked with the archival of this interaction is recorded. The number can be also described as epoch, which signifies the point from which the registered entity is authorized to receive events conforming to the template. Once a template event has been archived, RDS issues a notification. The notifications allow a subscribing entity to keep track of the template events while facilitating error detection and correction. Upon receipt of the notification, the subscribing entity confirms the reception of the corresponding template event.

When an entity reconnects to the broker network after failures, the entity retrieves the template events that were issued and those that were in transit before the entity leaving. After the receipt of the recovery request, RDS scans the dissemination table starting at the sync related with the entity and then generates an acknowledgment-response invoice event outlining the archival sequences, which the entity did not previously receive. Accordingly, the missing events are provided to the receiver.

In addition to this, a reliable mechanism for Web Service handler execution environment is built on the top of the reliable messaging that NaradaBrokering provides. The distributed Web Service handler mechanism is able to repeat the execution of a specific handler in the situation of a failure. The decision of a failure is made when the response is not received from a distributed handler. There can be several reasons behind being unsuccessful to get a response. The communication link may be broken as well as the handler may not successfully process the message because of either an error or crash. The distributed Web Service handler mechanism checks the possibilities by sending the message several times to its destination. In each attempt, it waits for a specific amount of time. This duration is either assigned or calculated by the system. After having several unsuccessful attempts, the message processing may switch to a replica if it exists. As it is discussed previously, handlers can populate their replicas to improve availability and reliability.

For the reliable messaging benchmark, two HP DL 380 G7, 2 x Xeon Six Core, 2.93 GHz, and 48 GB memory physical machines are utilized. The machines are virtualized to create four 4-core and 16 GB memory machines and one 8-core 32 GB memory machine. These machines are connected to each other via LAN and share a common storage system. Virtual machines use Windows Server 2008 R2 64-bit operating systems. The cost of reliable mechanism of the messaging for the distributed handlers is shown in Figure 4. The cost contains the time of reliability procedures to send the tasks to the distributed Web Service handlers or receive the responses back. The time for the handlers' executions and the time for the messaging are excluded to illustrate only the reliability cost for varying message sizes. The figure shows that the message size does not affect the cost of the reliability of the messaging very much. The cost is very reasonable when the reliability is a necessity for the distribution.

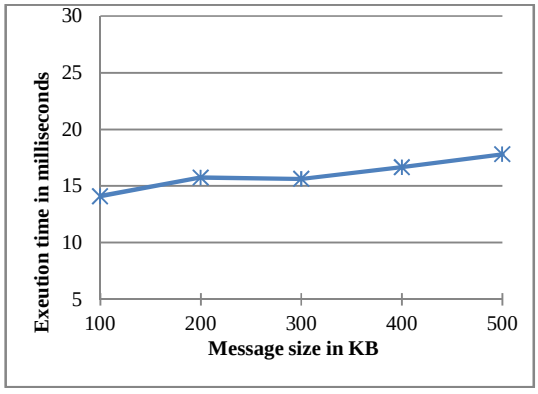

Figure 4. The cost of reliability mechanism of the messaging for the distributed handlers

V. MESSAGE SECURITY

Security is one of the important issues for the computing systems. The very critical data can be seen or altered by an unauthorized person. This is increasingly important if the data is transferred through the network, which is more vulnerable environment.

The local computing is not exposing its data to the outside world very much. In contrast, this is not the case for the distributed computing. The computation is shared between the nodes which may physically disperse in the distributed environment. The transmission of the data among the nodes may expose the critical information to the dangerous vulnerabilities. Hence, the transportation channels must be secured in addition to the security of the computing entities.

NaradaBrokering, which is utilized for messaging, has a security framework that is able to support secure interactions between the distributed handlers [21]. The security infrastructure consists of Key Management Center (KMC), which provides a host of functions specific to the management of keys in the system. At the same time, KMC incorporates with an authorization module to manage the usage of the messaging. KMC also stores the entities public keys.

NaradaBrokering has an authentication mechanism for the publishers and subscribers, which are the computing nodes for the distributed handler execution. For the authentication, publisher or subscriber sends its signed request by using private key. Every topic has access control list which authorizes the subscriber. Similarly, an access control list exists for the publishers. After verification of signature, entity is permitted to be accessed by the publisher or subscriber according to the relevant access control lists.

The message traveling between the computing nodes is described in Figure 5. It contains a unique id, properties and a payload. Unique message id is a distinctive name for a message. The handler execution mechanism may host many messages being executed in a moment. Hence, an identifier is a necessity to achieve the correct executions; a Universally Unique Identifier (UUID) generated id is assigned to every message. The generator assures that there won't be the same id in the system. Thus, the design gives enough guarantees that the message executions are not blended.

```
<context>
 <id>4099d6dc-0b0e-4aaa-95ff-2e758722a959</id>
  <properties>
   <encKey>abcdef</encKey >
    ….
 </properties>
 <payload>
   ….
 </payload>
</context>
```

Figure 5. The message format for distributed Web Service handlers

The second important part of the message format is the properties section. This part conveys the required additional information for the computing nodes. The information can be specific to a handler as well as generic for all handlers. There is a property that contains a key for the encryption. It is a session key which is created for a single message. However, the session key can be utilized to send a group of messages to a distributed handler for a period of time. The payload containing the original message is encrypted by this key before sending to its destination to keep the message integrity intact.

In many distributed design, secure data transmission is not discussed, the models rely on the existing security technology such as Secure Socket Layer (SSL). Kemathy at al. investigates component base solution for XML messaging [22]. Ammari at al. provides architecture securing XML messages by encrypting flagged XML parts each with different type of encryption depending on data sensitivity and importance level defined [23], Figure 6 demonstrates the secure messaging for the distributed handlers. The XML based message of the distributed handlers is partially encrypted, only the payload. Since encryption via asymmetric key performance is worse than the symmetric key encryption [24], Advanced Encryption Standard (AES) symmetric key encryption algorithm is used to encrypt the payload. A 256 bit session key is created for each message and passed within the message to the other computing node for decryption. The sender encrypts the session key with the 2048-bit public key of the receiver to present the confidentiality. The related public key is provided by the KMC. RSA algorithm is used for the key encryption. Hence the only node, which has the correct private key, can decrypt the session key to get the payload.

When the subscriber receives the message, first of all, it decrypts the session key carried within "encKey" tag with its private key. Then, the session key is used to decrypt the payload to get the original message.

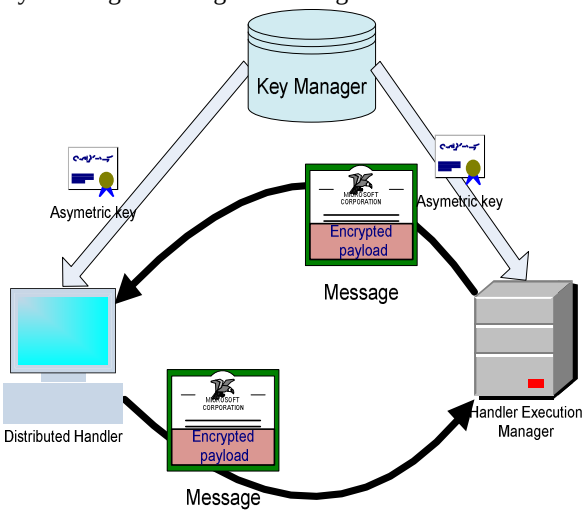

Figure 6. Security mechanism for a distributed handler

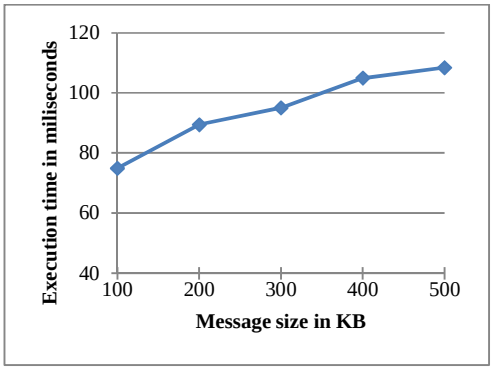

Figure 7. The cost of the security mechanism of the messaging for the distributed handlers

The benchmark showing the cost of the aforementioned security mechanism for the messaging is performed in the same environment with the reliability benchmark, discussed Section IV.B. Figure 7 shows the cost for varying payload sizes. The usage of the symmetric key encryption provides reasonable execution time. Even though the reliability offers better results, the cost of security does not grow exponentially for the increasing message size.

VI. CONCLUSION

While the distribution of Web Service handlers provides many advantages in terms of scalability, availability and performance, the environment necessitates reliability and secure messaging. The instruments, explained in this paper, for the secure and reliable handler distribution and the support tools of the utilized messaging broker grant the necessary reliability and messaging security for this environment. The benchmark results show that the costs originating from the utilized instruments are acceptable. The replication of the handlers contributes the execution during failures. In short, the design of the distributed execution with the security and reliability offers a satisfactory environment for Web Service handlers.

REFERENCES

[1] J. Hagel and J.S. Brown, “Your next IT strategy,” Harvard Business Review, 79 (10), pp. 105-113, 2001.

[2] P. Bzoch and J. Safarik, "Security and reliability of distributed file systems," in IEEE 6th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS 2011). vol.2, pp.764-769, Sept. 2011, doi: 10.1109/IDAACS.2011.6072873

[3] M. Lei, S. V. Vrbsky, and Z. Qi, “Online grid replication optimizers to improve system reliability,” in IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007), pp.1-8, March 2007.

[4] K. Birman, R van Renesse, and W Vogels, "Spinglass: secure and scalable communications tools for mission-critical Computing," in International Survivability Conference and Exposition (DARPA DISCEX-2001), CA, June 2001.

[5] C. M. Jayalath and R. U. Fernando. “A modular architecture for secure and reliable distributed communication,” in Proceedings of the Second International Conference on Availability, Reliability and Security (ARES07), pp. 621-628, 2007, Washington, DC. DOI=10.1109/ARES.2007.7 http://dx.doi.org/10.1109/ARES.2007.7

[6] Web Service Security (WS-Security), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss/, <retrieved: 03, 2012>.

[7] Web Service Reliable Messaging (WS-ReliableMessaging), http://public.dhe.ibm.com/software/dw/specs/ws-rm/ws-reliablemessaging200502.pdf, <retrieved: 03, 2012>.

[8] B. Yildiz, G. Fox, and S. Pallickara, “An orchestration for distributed Web wervice handlers,” in International Conference on Internet and Web Applications and Services (ICIW08), pp. 638-643, June 2008, Athens, Greece

[9] B. Yildiz, “Distributed handler architecture,” Ph.D. Dissertation. Indiana University, Bloomington, IN, USA. Advisor: Geoffrey C. Fox. 2007.

[10] B. Yildiz and G. Fox, "Measuring overhead for distributed Web Service handler," in Proceedings of 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT 2010), pp. 566-570, July 2010.

[11] H. Zo, D. Nazareth, and H. Jain, "Measuring reliability of applications composed of Web Services," in Proceedings of 40th Annual Hawaii International Conference on System Sciences (HICSS '07), pp. 278- 288, 2007.

[12] J. D. Musa, "Software reliability engineering," McGraw-Hill, New York, NY, 1999.

[13] A. Arsanjani, B. Hailpern, J. Martin, and P. Tarr, “Web Services: promises and compromises, ”ACM Queue, 1 (1), pp. 48-58, March 2003.

[14] J. Zhang and L.-J. Zhang, “Criteria analysis and validation of the reliability of Web Services-oriented systems,” in Proceedings of the IEEE International Conference on Web Services (ICWS'05), Orlando, Florida, July 2005.

[15] A. Helal, A. Heddaya, and B.K. Bhargava, "Replication techniques in distributed systems," Kluwer Academic Pub. 2002, Volume 4, pp. 61-71, DOI: 10.1007/0-306-47796-3_3.

[16] P.A. Lee and T. Anderson, "Fault tolerance: principles and practice," Springer-Verlag New York, Inc. Secaucus, 1990.

[17] W. Zhao, P.M. Melliar-Smith, and L.E. Moser, "Fault tolerance middleware for cloud computing," in IEEE 3rd International Conference on Cloud Computing (CLOUD 10), pp. 67-74, July 2010.

[18] P.T.T. Huyen and K. Ochimizu, "Toward inconsistency awareness in collaborative software development," in 18th Asia Pacific Software Engineering Conference (APSEC), pp. 154-162, Dec. 2011.

[19] S. Pallickara and G. Fox, “NaradaBrokering: a distributed middleware framework and architecture for enabling durable peer-to-peer grids,” *In* Proceedings of the ACM/IFIP/USENIX International Conference on Middleware (Middleware '03), pp. 41-61, 2003.

[20] S. Pallickara and G. Fox, "A scheme for reliable delivery of events in distributed middleware systems," in Proceedings of the IEEE International Conference on Autonomic Computing (ICAC'04), New York, NY, pp. 328-329, May 2004.

[21] S. Pallickara, M. Pierce, G. Fox, Y. Yan, and Y, Huang, "A Security framework for distributed brokering dystems", Available from http://www.naradabrokering.org, <retrieved: 03, 2012>.

[22] K. Komathy, V. Ramachandran, and P. Vivekanandan, “Security for XML messaging services: a component-based approach,” Journal of Network and Computer Applications, Vol. 26, Iss. 2, pp. 197-211, April 2003, DOI=10.1016/S1084-8045(03)00003-1 http://dx.doi.org/10.1016/S1084-8045(03)00003-1.

[23] F. T. Ammari and J. Lu, “Advanced XML security: framework for building secure XML management system (SXMS),” In Proceedings of the Seventh International Conference on Information Technology: New Generations (ITNG '10), Washington, DC, pp. 120-125, 2010, DOI=10.1109/ITNG.2010.124 http://dx.doi.org/10.1109/ITNG.2010.124.

[24] C. Narasimham and J. Pradhan,“Evaluation of performance characteristics of cryptosystem using text files”, Journal of Theoretical and Applied Information Technology,Vol. 4, Iss. 1, pp. 56-60, 2008.