Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 271 – 278

ISSN 1896-7094

Introducing fault-tolerance and responsiveness in web applications using SREFTIA

Muaz Niazi[1] H. Farooq Ahmed[1] Arshad Ali[1]

[1]NUST Institute of IT, St 9, Chaklala Sch 3, Rawalpindi, Pakistan
{muaz, arshad.ali, farooq.ahmad}@niit.edu.pk

Abstract. We present a framework for adding fault tolerance and responsiveness to an existing web based application using transportable information agents. This framework is presented in a step by step manner and uses several agents to introduce fault-tolerance in a system. We also show simulated results for an actual application where we introduce fault tolerance and responsiveness and demonstrate a reduction in down time as well as an improvement in the responsiveness.

Keywords: Fault Tolerance, multi-agent system, data source, multiple, distributed transactions, request reply

Every successful web application needs to scale. With success, comes business. And with business, comes the need for enterprise application architecture. However, although a very common problem, it is fairly hard to re-factor an existing architecture to an enterprise application. In this paper, we demonstrate the use of SREFTIA (Short Response and Enterprise Fault Tolerance using Transportable Information Agents) in moving a basic dynamic data centric web application to an enterprise level application with enhanced fault-tolerance and shorter response to the end users. The framework improves fault-tolerance and application responsiveness during faults as well as fault discovery and correction

The key question that we try to answer is “How can a small business application be upgraded to a full-scale enterprise data-centric application?”

1 Introduction

1.1 Challenges in Enterprise applications

eCommerce and distributed business applications, are quite frequently, user dependent and they simply cannot afford to go down. A challenge for the typical web based businesses is the process of upgrading from a small scale application to a large-scale data centric enterprise application. In our work, we present a sample web-based application and demonstrate how to upgrade the application in different steps eventually leading to an enterprise application. The entire idea is that when the typical Jane Doe clicks on a link on her web portal, she does not expect to see an exceptional situation or the dreaded “Internal Server Error” for a web application in addition to an extending wait time. Our framework based on agents, which are automated pieces of software, offers a logical solution to this problem. The idea is to use a hetereogenous multi-agent system to dynamically discover path availability and to ensure that the correct path is selected to each data source. We demonstrate using a simulation

based on to the use of Transportable Information Agents [14] to provide not just fault tolerance in but also a short response time in case of system faults including complete path failure scenarios.

1.2 Related work:

Although a lot of work has been done on fault tolerance in general and some specific to multi-agent system, the focus of the work has typically been on improving the fault tolerance capability of the agents and the multi-agent system. In contrast, our focus is on giving a multi-agent system architecture which can be used to improve the responsive in failure circumstances as well as improved fault tolerance. This framework, if applied to an already existing web application, will provide for the following:

- Fault tolerance in multiple redundant paths to the data source(s)
- Shorter response time and lesser down time in case of system faults

The key innovations in our work are

- A fault tolerant architecture using TIAs
- A step-by-step guideline for application of the architecture to re-factor an existing system for reduction in fault tolerance and response time during faults.

[1] describes a proposal for modeling fault-tolerant systems using an agent-based approach. Fault tolerance mechanisms are provided for making the system dependable. [2] proposes a proactive self-healing system that monitors, diagnoses and heals its own internal problems using self-awareness as contextual information. [3] MAFTS is a Multi-agent based fault-tolerant system for fault-tolerance in Multimedia environment and the key idea is error detection using polling but it does not involve business systems and transactional support. In [4], a strategy to implement fault-tolerance in a multi-agent system is proposed which is based on re-construction of agents based on antecedence graphs however they do not deal with business systems which do not require reconstruction of existing agents and instead need a focus on being responsive to users. In DARX [5], adaptive fault tolerance is provided in a multi-agent system; again the focus is on fault-tolerance in agents and the idea here is based on adaptive replication in computations based on agent importance, network load and mean time between failures. In [6], the approach is interesting as it also uses mobile agents for fault tolerance and security in eCommerce systems. And similar to our work, it uses distributed transactions. However, first of all, the goal in our system is shorter response time instead of only fault tolerance. Secondly, the focus is on using only one particular technology OMG for the distributed transactions. Our work, although similar, is not limited to one technology and although currently implemented in Java, it is portable across systems with completely de-coupled servers, each running different types of operating system and software assuming a compliant middleware is installed. E.g. .NET framework instead of Java runtime. In addition, again the approach is to implement fault tolerance in agents vs. using the multi-agent system to implement fault-tolerance. In addition, [7] deals with inaccessibility of agents in certain situations and creates agents called doubler and stand-in agents which maintain accessibility. [8] focuses on heterogeneous multi-vendor environment but still relies on J2EE solutions in contrast to our work. [9] provides a similar fault-tolerance mechanism for eCommerce however the focus here is not on responsiveness. Focus of [10] is again on developing a fault tolerant multi-agent system. Another interesting approach is [11] whose focus is on fault-tolerant agent-based cluster computing. The work [12] is focused on improving fault-tolerance in a multi-agent system. Transportable Information Agents are described in [13].

1.3 Related work on Agents, Multiagent systems and related concepts

There is much argument over the exact definition of an agent but the reader can consult some good references such as [14, 15]. [14] mention that

"While most agents are static in that they exist as a single process or thread on one host computer, others can pick up and move their code and data to a new host in the web, where they then resume executing". Although agents can be mobile however not all agents need to be mobile [15]. Essentially agent mobility is a form of code migration. Transportable Information Agents can be used to discover network status.

1.4 Structure of paper

The rest of the paper is structured as following:

To illustrate the usefulness of the SREFTIA add-on framework, we first present the problem. Next we describe the framework architecture. Finally we describe the algorithms, test cases, experimental setup and results.

2 The problem and the solution

In this section, we present the sample problem and then the algorithms associated with it.

2.1 Sample problem

First of all, let us examine a typical data centric web application. This application, shown in Figure 1 below, can handle only a basic level of network traffic and in case of any faults, there is no redundancy available. On the left side, we have different users connecting via the web to the web server. This server is connected to a database server either hosting the database directly or connecting to a database on another machine. As such, with an increase in the customers, this application is not going to scale. And if any connectivity goes down between the various links, the entire application can become either slow or in the worst case, completely crash.

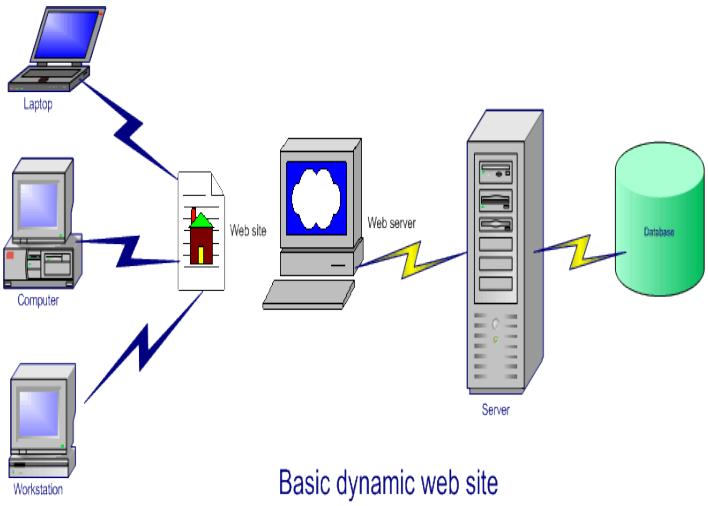

Fig. 1. Application to be made fault-tolerant

2.2 Solution to the problem

The solution offered by SREFTIA has to be implemented in two phases

Phase1:

First of all, redundancy needs to be added to the system. This means adding additional hardware as shown in the Figure 2.

Phase 2:

In the second phase, the SREFTIA framework is added to the architecture. SREFTIA framework would require an agent toolkit on the systems where agents are to exist.

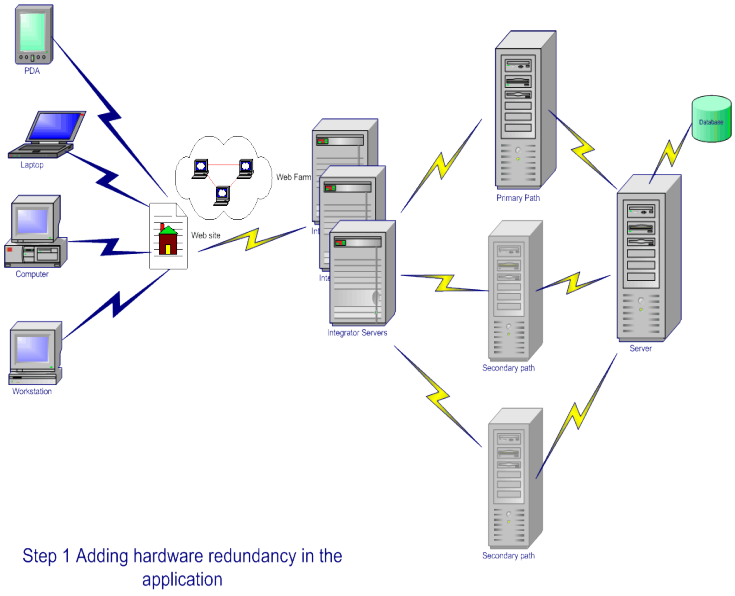

Fig. 2. Making the architecture stronger by adding additional hardware

After the agents have been added, we see our application now looks as below in Figure 3.

Please note that the redundancy that has been added is realistic in the sense that from the integrator servers, there are multiple paths to the data server connected to the database. However only one of these paths is the primary path. Typically selection of such a path can be based on network costs/hosting costs etc. In addition, there can be one or more servers on the paths. The messaging can actually be executed using a message infrastructure such as Websphere MQSeries. The other paths are there for fault tolerance. In case of failure, these paths may be used to channel the transactions.

Now, the details and functions of each of these agents are given below:

2.2.1 Transaction agent

This is an agent which is a TIA and moves from the transaction router agent to the data transaction agent by moving from one server to the other along its assigned pat path. Transaction agent has a payload of the transaction, which can be of two types:

Inquiry

In case of an inquiry, if a transaction fails, it can be retried by some other transaction agent.

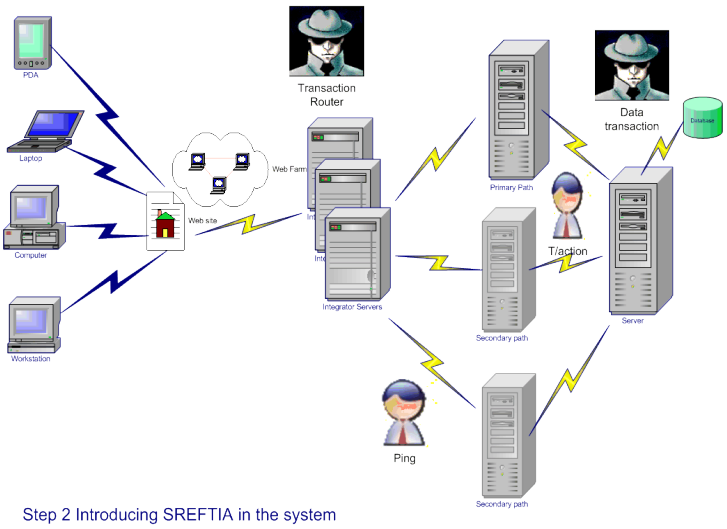

Fig. 3. Finally adding the agents of the SREFTIA framework to the system

Update

If it is an update, there is risk associated with retrying. As such, it does not make sense to retry. However, an entry can be written in the transaction log so that a human user can check the log at the end of the day for such transactions and ensure the execution of the same at the back-end.

2.2.2 Path Observer or ping agents

These are actually a special type of transaction agents which do not perform any data transaction. They are only there to ensure availability of a path; Each path consists of multiple servers connected to each other. There is a path observer agent assigned to each path which moves back and forth along the entire path and reflects the state of the path. The time for the ping to be initiated is based on a two prong approach; Ping timeout can be determined depending on whether the path was previously discovered to be available or not. In case of unavailability, we can have a shorter period whereas in case of previous availability, the ping can have a longer period so as to be helpful but not becoming a communication load itself on the system.

2.2.3 Transaction router Agent

This is the most important agent in the system. It keeps tracks of other agents of the system as well as information about the availability of the paths. It takes in transactions and puts them in a queue and notifies/creates a transaction agent. After the result comes back, it gives back the result back to the user. If a path is discovered to be down, it initiates the ping agent and sets a shorter ping period to ensure that the path is discovered to be available as soon as possible.

2.2.4 Data transaction agent

This agent is at the data server end and actually executes the transactions and returns back the results by sending the transaction agent back with the result of the transaction as a payload.

2.3 The algorithms

Pseudo code for the algorithms used in SREFTIA is described below.

2.3.1 For the Ping and transaction Agent

It runs in Forward role only and goes from one end to the other. It is re-sent back by the data transaction agent when it reaches that end.

Regular vs. Path down Ping

```
If (not Path)
        Timeout = small
Else
        Timeout = large
```

Work/thread function for the ping agent

```
Sleep (Timeout)
Go (Forward, null)
```

Ping

```
Go(Direction, null)
{
        While (not (Last machine)
        {
         If (available(next machine))
                Move(Next machine)
        Else
                Go(Back, null)
        }
}
```

2.3.2 For the transaction router

Transaction added to queue, agent notified

```
Do(transaction)
{
For each path starting with primary path
If (Path.isAvailable)
        Use path
Else
        Start ping agent on path
If(Path)
        TransactionAgent.Go (Forward, transaction)
Else
        Give feedback to user: Application down,
}
```

2.4 Experiments

The simulation experiments were conducted using ® Matlab. The simulation is based on a range of fault frequencies from 1 fault per 100 minutes to 100 faults per 100 minutes. In addition, the following graphs demonstrate the effects of the enterprise application with three different fault ranges, (2-5, 2-6 and 2-16 sec).

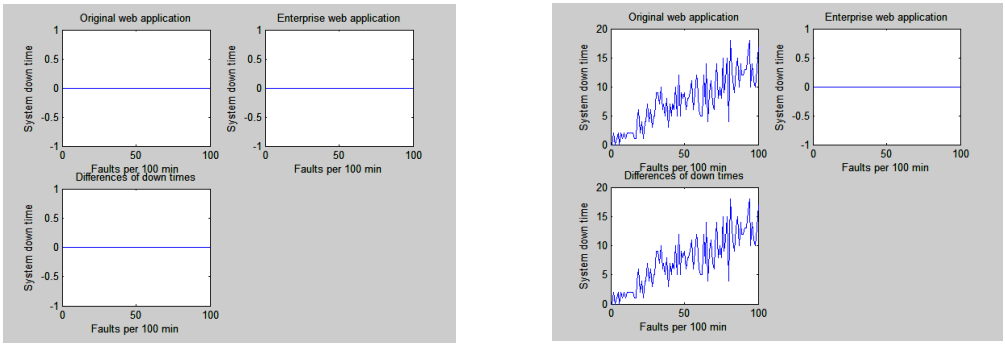

Fig. 4. (a) Faults of length 2-5 sec and (b) faults of 2-6 sec, time out per path = 5 sec

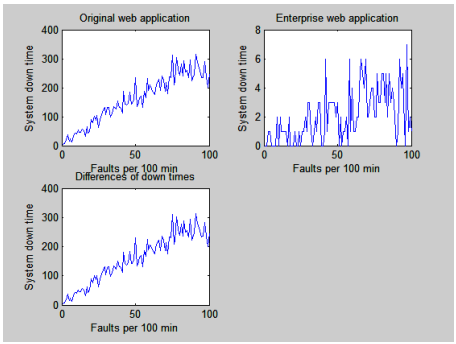

Fig. 4. (c) Faults of length 2-16 sec, time out per path = 5 sec

2.5 Discussion

The simulation is based on certain parameters. In Fig 4a, we see that the faults are overall smaller than 5 sec, a time which can be handled by both the basic application as well as the web application. In Fig 4b, we see a fault length of 2-6 sec, which means that there may be certain faults which will give a downtime to the basic web application but will be handled well by the enterprise web application. In Fig 4c, we see the range of faults from 2 to 16 seconds, which means that there may be certain faults which cannot be handled by even the enterprise application and thus we do see some downtime.

3 Conclusion & future work

In this paper, we have presented a framework using a set of agents which provides both a short response time as well as an improvement in overall fault tolerance in the enterprise system. In contrast to other approaches, our approach is more of an add-on solution to existing designs based on distributed applications with distributed data stores. In addition, we have presented example scenarios involving converting a typical data centric web application to an enterprise application relying on the availability of multiple paths. In the future, we expect to formalize the framework and to use adaptive agents in the SREFTIA to improve fault tolerance.

4 Acknowledgements

This work on a generalized distributed system would not have been possible if I had not gotten the basics of specialized distributed systems right while developing enterprise applications with my ex-colleagues including Dave Marteney, Dan Chartier, Karen Low, Tobey Johnson, Eugenio Arroyo, Ann Galdos, Donna Boyko and the rest of the team.

References

1. Modeling Fault-Tolerant Systems using BDI Agents (2003) Felipe Rech Meneguzzi Proceedings of the 2nd Workshop on Theses and Dissertations on Dependable Computing.
2. Jeongmin Park, Giljong Yoo, Eunseok Lee: *Proactive Self-Healing System based on Multi-Agent Technologies,* sera, pp. 256-263, Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05), 2005.
3. Eung-Nam KO: *A Multi-Agent based Fault Tolerance System for Distributed Multimedia Object Oriented Environment: MAFTS,* sera, pp. 122-129, Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05), 2005.
4. M. Masud Khokhar, Aamer Nadeem, Omer Mansoor Paracha: *An Antecedence Graph Approach for Fault Tolerance in a Multi-Agent,* mdm, p. 137, 7th International Conference on Mobile Data Management (MDM'06), 2006.
5. Olivier Marin, Marin Bertier, Pierre Sens: *DARX - A Framework For The Fault-Tolerant Support Of Agent Software,* issre, p. 406, 14th International Symposium on Software Reliability Engineering, 2003.
6. Hartmut Vogler, Thomas Kunkelmann, Marie-Louise Moschgath: *An Approach for Mobile Agent Security and Fault Tolerance using Distributed Transactions,* icpads, p. 268, 1997 International Conference on Parallel and Distributed Systems (ICPADS'97), 1997.
7. Michal Pechoucek, Michal Dobisek, Jiri Lazansky, Vladimir Marik: *Inaccessibility in Multi-Agent Systems,* iat, p. 182, IEEE/WIC International Conference on Intelligent Agent Technology (IAT'03), 2003.
8. Milovan Tosic, Arkady Zaslavsky: *The External Fault-tolerant Layer Supporting Multi-agent Systems from Different Vendors,* iat, pp. 435-438, IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2005.
9. Prithviraj Dasgupta: *A Fault Tolerance Mechanism for MAgNet: A Mobile Agent based E-commerce System,* International Conference on Internet Computing, pp. 733-739, 2002.
10. Sanjeev Kumar and Philip R. Cohen: *Towards a Fault-Tolerant Multi-Agent System Architecture,* Number CSE-99-015-CHCC, November 23, 1999.
11. Kam Hong Shum: *Effective fault tolerance for agent-based cluster computing,* Journal of Systems and Software, **48**(3), pp. 189-196, 1999.
12. Andraz Bezek and Matjaz Gams: *Increasing Fault-Tolerance of Multi-Agent Systems* Informatica (Slovenia), **27**(4), pp. 417-424, 2003.
13. Daniela Rus, Robert Gray, David Kotz: *Transportable Information Agents,* Readings in Agents, Bp 283-291 1997.
14. *Readings in Agents,* Edited by Michael N. Huhns & Munindar P. Singh 1997..
15. *An introduction to MultiAgent systems,* Michael Wooldridge 2002.
16. `http://jade.tilab.com/`