Client-Transparent Fault-Tolerant System for SOAP-based Web Services

Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka

Abstract

This paper describes a new scheme for providing client-transparent fault-tolerance for SOAP (Simple Object Access Protocol) based web services. We have designed our scheme in the context of a distributed object system called FAWS, a portable fault-tolerant system that guarantees the full availability of SOAP-based web services while existing fault-tolerant systems for web services do not provide fault-tolerance for transparent handling of requests whose processing was in progress when the failure occurred. FAWS provides a client-transparent mechanism for providing fault-tolerance in such events. FAWS consists of a FT-Admin, FT-Monitor, FT-Front, FT-Detector and a primary server with redundant secondary servers and uses message logging. The unique features it has, overcome some drawbacks of the existing implementations. FAWS has the potential to be a powerful tool for e-business critical web services due to its ability to guarantee full availability of service in the presence of failures.

Key Words:
Fault Tolerant System, Replication, Client Transparency, SOAP, Web Services, Availability, Fault Tolerance, Failover

1. Introduction

Web services are software-powered resources or functional components whose capabilities can be accessed at an Internet URI (Uniform Resource Identifier). Standards-based web services use XML to interact with each other, which allow them to link up on demand using loose coupling. Web services often suffer from long response times or temporary non-availability. For some classes of web-based applications, e.g. online banking, stock trading, reservation processing and shopping, such outages are unacceptable. Fault-tolerance techniques can be used to address these problems and to get a better solution.

Recently, SOAP (Simple Object Access Protocol) has become the most popular technology to develop web service applications. But SOAP-based web services require fault-tolerance support when it is used for critical applications such as those mentioned above.

The major problems [3][4] faced by clients who use SOAP to access a web service are service non-availabilities and request resending. When a client sends a request and if the server fails while the request is being processed, the client will be disconnected and will not get the desired response. Then such a client has to resend the SOAP request to access the service. A fault-tolerant system for SOAP-based web services can prevent such undesirable situations.

Among the fault-tolerant techniques [6][12], the client-transparent server replication technique is the most suitable to address the non-availabilities in SOAP-based web services. Both the passive and active replication mechanisms can be easily implemented in such a system. Client transparency is achieved by publishing only one server address to access the web services. The backup servers and the fault handling process are not visible to client.

FAWS, a portable fault-tolerant system, is developed to address the non-availabilities of the SOAP-based web services in a client-transparent manner to provide the full availability of service in the presence of server and service failures. This system is associated with a fault detector that detects server and service failures. FAWS routes SOAP requests to a standby secondary server when a failure is detected in the primary server by the fault detector.

FAWS has the ability to operate on different operating system environments. FAWS is based on distributed object architecture and hence it consists of several components that operate independent of each other. This guarantees the operation of the fault-tolerated system in the presence of single component failure in FAWS. Also distributed architecture lowers the risk of failure of the whole system, which can lead to chaos. Each component of FAWS is capable of operating in either Linux or Windows environments.

Unlike other implementations [1][3][6], when FAWS is applied to existing web services, only a minor modification to the WSDL document for each web service is needed.

The graphical user interface of FAWS eases the configuration and monitoring of the target system. The
user interfaces hides the complexity of FAWS from the users. The front end of FAWS acts as the web server and receives SOAP requests. When a request is received, it is forwarded to the current primary web server and the response from the server will be sent back to the respective client. Under FAWS, clients do not have to send their SOAP requests more than once to access a web service. Front end logs each received request before it is sent to the primary server and this avoids the clients to resend the request in case of a primary failure while the request is being processed. Clients use only the address of the front end to access web services and hence the underline primary and secondary servers are not visible to clients.

This paper is organized as follows. Section 2 describes the background of fault-tolerant techniques and section 3 describes the architecture of FAWS. Operation and implementation of FAWS is described under sections 4 and 5 respectively. Section 6 presents related work with two implementations in the same domain. Section 7 and 8 presents the FAWS performance evaluation and future work respectively.

2. Background

In this section we present a brief introduction to fault-tolerant scheme for web services.

Fault-tolerant schemes increase the availability of web services by providing the ability to service new requests following server or service failure. Such schemes may also increase the reliability of the service by ensuring that even requests that are in progress at the time of server failure will be handled correctly.

Any fault-tolerant scheme is either stateless or stateful depending on the mechanism used by the scheme to preserve the state of a received request.

Stateful fault-tolerant schemes preserve the state of the received requests in presence of web service or server failure. On the other hand, stateless fault-tolerant schemes do not preserve or repair the state of a received request and make no attempt to handle the requests that are being processed at the time of server failure.

There are several approaches to develop stateful fault-tolerant schemes. These include active replication [2], message logging and checkpointing [4].

Active Replication: Active replication requires lot of resources. In the context of web services it means the need for redundant web servers. When the fault-tolerant system receives a request from a client, it is sent to all server replicas and each replica computes independently the result and sends it back to the fault-tolerant system, which, in turn, passes the first arrived result to the client and filters out the others. Any web server failure is transparent to the client.

Message Logging: In this approach, each request received by the fault-tolerant system from clients is logged. If the web server fails while a request is being processed, the logged request is then replayed to an alternate server. A server failure under this approach is also transparent to the client.

Checkpointing: An up-to-date copy of the server state is maintained on a standby backup or in stable storage. For transparent failover, the checkpointed state must be recovered to a replica server with the same identity as the failed server.

Web service is typically provided using HTTP over TCP. The TCP connection has state that must be preserved. Many Internet services are provided using three-tier architecture, consisting of (i) a client web browser (ii) one or more replicated front-end servers (e.g. Apache web server) (iii) one or more back-end servers (e.g. Database server). In the context of web services, in general the front-end servers are considered to be stateless since back-end servers manage most of the relevant state. Therefore most [2][4] fault-tolerant schemes for web services simply provide standby backup servers that can start processing new requests in place of a failed server. Passive replication is a common approach to develop a stateless fault-tolerance scheme.

Passive Replication: A fault-tolerant system that uses passive replication consists of one primary web server and one or more backup servers. This approach needs fewer resources than in active replication. Each request received by fault-tolerant system is sent to the primary server for processing. If the primary server fails while processing a request, then the fault-tolerant system simply discards the request that was being processed and selects one of other backup servers as the primary server. Under this approach, server failure is not transparent to client.

3. Architecture

FAWS uses a combination of passive replication and message logging to preserve the state of received requests and to achieve client transparency. Passive replication does not require lot of resources (web servers) in FAWS. Therefore FAWS is capable of providing fault-tolerance for web services with fewer resources than that in other system [3] [6].

Figure 1 presents an overview of FAWS architecture. This shows the high-level object architecture of FAWS. The interaction of four major components of the system; FT-Front, FT-Admin, FT-Detector and FT-Monitor with the client and primary and backup servers can be seen. Although FT-Admin and FT-Monitor are shown as two separate components in Figure 1, they run as a single process in the implemented system. Each of these components works independently and communicates with several other components.
3.1 FT-Front

FT-Front is the front end of FAWS. This is the only component of the system that interacts with both the client and the primary web server.

FT-Front listens for SOAP requests on a specific port set by the administrator. All the web services under FAWS are published with the IP address and port number of the FT-Front. Then client sees the FT-Front as the web server that provides the web services. This mechanism hides the primary and redundant backup servers from the client. The state management of clients is also handled by the FT-Front.

FT-Front uses RMI to communicate with the FT-Admin. FT-Admin starts up the FT-Front by providing configuration data such as the IP address of primary web server, maximum number of resends per request etc. Also FT-Front has the ability to failover to a new primary server dynamically when FT-Admin notifies. This is possible due to the maintained message log.

3.2 FT-Admin

FT-Admin could be considered as the core of the system. FT-Admin can be used to manage the fault-tolerate system as a single entity and by managing other components as if they were running on a single server.

FT-Admin provides two basic services. They are Replication management and Configuration Management. Replication management is responsible for maintaining the replicated servers and failover operations. Configuration management service is responsible for system initialization and changes.

Main sub-components of the FT-Admin are FT-Group, FT-Manager and FT-Admin Interface. FT-Group defines the group of servers, which exists in the fault-tolerant system. This can be used to create a FT-group object that represents the whole fault-tolerant system. FT-Manager defines the methods, which are used to start and initialize other remote components in the fault-tolerant system. Then it defines the methods used to update the system components when the system changes have been made. FT-Admin Interface provides the interface for the remote methods called by FT-Front and FT-Detector.

FT-Admin is one of four main components of FAWS and it could be considered as the core of the system. FT-Admin can be used to manage the fault-tolerance system as a single system and to manage applications as if they were running on a single server. It always communicates with FT-Detector and FT-Front in order to provide uninterrupted service.

Basically FT-Admin provides two services. They are Replication management and Configuration Management. Replication management is responsible for maintaining the replicated servers and failover operations. Configuration management service is responsible for system initialization and changes.

Main components of the FT-Admin are FT-Group, FT-Manager and FT-Admin Interface. FT-Group defines the group of servers, which exists in the fault-tolerant system. This can be used to create a FT-group object that represents the whole fault-tolerant system. FT-Manager defines the methods, which are used to start and initialize other remote components in the fault-tolerant system. Then it defines the methods used to update the system components when the system changes have been made. FT-Admin Interface provides the interface for the remote methods called by FT-Front and FT-Detector. This Interface declares the methods that can be invoked by other components remotely in order to provide the internal communication among system components.
3.2 FT-Detector

FT-Detector detects software and hardware failures and notifies them to FT-Admin. Two classes of failures are detected: (i) when a machine has failed, (ii) when a process has failed. We need to distinguish those two failures.

FT-Detector consists of two sub components. One component called \textit{ft\_process} checks whether a service is failed. The other component \textit{ft\_machine} will check for hardware failures.

If a web server is up and running, it will listen on a specific port. By checking whether the server process actually listens on that port, \textit{ft\_process} decides whether the web server process is up or down. To detect machine failures, ICMP protocol is used. ICMP echo requests will be sent by \textit{ft\_machine} to each server machine periodically. If a machine is failed, it will not echo that message. If \textit{ft\_machine} does not get a reply within a time-out period, it will resend an ICMP request and wait for the reply. If it does not get a reply to the second request, FT-Detector will decide that the particular machine is failed. Communication between FT-Detector and the FT-Admin is conducted using RMI.

3.3 FT-Monitor

FT-Monitor is the main graphical user interface of FAWS. It provides two major functionalities to system admin or the system user;
1. Provides facility to set the system configuration at the initial stage as well as at the run time, and
2. Provide system status and graphical representation of distributed FAWS components.

FT-Monitor communicates only with FT-Admin in order to take required information to display the system status, and it gives required information to FT-Admin in order to change system configurations. In addition to that, depending on the system status, it gives an alert to system user.

4. Operation

Figure 2 shows the operation of FAWS. This describes the operation considering one client for convenience, although FAWS is capable of handling multiple clients. Each step number corresponds to the numbers marked in Figure 2.
1. Client sends the SOAP request to FT-Front.
2. FT-Front accepts the client and creates a separate thread named client-thread to handle the client. Starts the \textit{client-thread}.
3. Client-thread running in FT-Front logs the message.
4. Client-thread gets IP address of the current primary server and forwards the request to it.
5. Primary server processes the request and sends back the response to the respective client-thread in FT-Front.
6. Client-thread sends the received response from the server to the client.
7. Client-thread removes the logged SOAP request and terminates.

Above steps described the operation of FAWS under faultless conditions. If any fault or failure is detected by FAWS, its operation deviates a little from above. Steps 8 to 13 describe the operation of FAWS under such a faulty condition.
8. FT-Detector detects a failure (either machine failure or web server process failure) in primary.
9. FT-Detector notifies FT-Admin about the failure.
10. FT-Admin selects one of the backup servers as the primary and notifies FT-Monitor to show the current status.
11. FT-Admin notifies the FT-Front about the new primary server. FT-Front changes its current primary address to the new address.
12. When the primary is changed by FT-Admin, FT-Front discards the requests sent to the failed primary and access the message log to get the logged messages, which have been sent to the previous primary.
13. FT-Front sends the acquired logged requests to the new primary. After that the system operation is similar to the operation described in steps 5, 6 and 7 respectively.

FT-Front is also capable of detecting primary server failures. This fault detection system associated with FT-Front is useful in case of a FT-Detector failure. Steps 14 and 15 describe what happens if FT-Front gets an exception while trying to access primary web server.
14. FT-Front gets an exception while sending a request to the primary server because it is failed.
15. FT-Front requests a new primary address from FT-Admin. This failure detection system associated with FT-Front guarantees the full operation of FAWS in case of a FT-Detector failure. After this step, system operation is similar to the operation described in steps 10, 11, 12 and 13 respectively.

5. Implementation

FAWS was implemented using Sun Java Development Kit version 1.4.2. JBuilder 9 was used as the Integrated Development Environment (IDE), on both Red Hat Linux 9 and MS Windows XP platforms. JBuilder 9 was the latest release IDE from Borland by the time that FAWS was implemented and it consisted of many features and wizards to facilitate coding. The four main components of FAWS were developed as separate modules.

FAWS has been implemented using Java coding convention [11]. We manage memory ourselves without waiting until garbage collector does the job. And we have minimized occurrence of exceptions.

Considering the technologies used to develop FAWS, risk assessment of the project indicated that the highest risk is the technology risk. Therefore each component had to be developed individually and tested in different stages in their development. Four basic prototypes of the main components have been developed to implement basic functionalities of the system. Java RMI was used for communication between components.

Since all four components were developed independently, component testing was done very often during their development. This had ultimately improved the overall efficiency of the implementation. The complete system has also been thoroughly tested.

6. Related Work

Many solutions have been developed to provide fault-tolerance in web services. We design FAWS architecture to minimize drawbacks in other solutions [1][3] and to minimize changes to be done in the web service architecture. Here we will look at some fault-tolerant solutions similar to FAWS.

FT-SOAP [3] is a client-transparent fault-tolerant system for web services and achieves fault tolerance by modifying the SOAP architecture.

They have introduced a new tag called web service group or WSG to the WSDL. This tag basically defines what the primary server is and what the backup servers are and so on. The client side SOAP engine will have a chance to inspect the WSG information if the invocation is failed. The client SOAP engine may try to send invocation to rest replicas in order till the invocation is successfully executed. If all replicas listed in the WSG could not response the request, then the SOAP engine returns a failure exception to client application.

In order to deploy a web service using FT-SOAP we have to make changes to SOAP architecture. We found that changing SOAP architecture will incorporate some overhead. So we design FAWS in such a way that we don’t have to do any modifications to SOAP architecture.

When deploying a fault tolerant web service by using FT-SOAP, there will be several overheads. According to their analysis, the system overhead comes from the logging mechanism, fault detection, and check pointing. FAWS try to minimize the overhead of logging requests by keeping requests in the memory.

Client-Transparent Fault-Tolerant Web Service [1] is another solution that provides recovery of requests that were partially processed. The system is based on a standby backup server and simple proxies. This system uses error-handling mechanisms of TCP to (i) multicast requests to the primary and backup as well (ii) to reliably deliver replies from a server that may fail while sending the reply.

The system is client-transparent. Clients communicate with a single server address, which is referred to as the advertised address, and are unaware of server replication. The advertised address consists of an IP address and a TCP port number. At the TCP/IP level, all messages sent by clients have the advertised address as the destination and all messages received by clients have the advertised address as the source address, regardless of the state of the system and the replicated servers. The primary and backup hosts are connected on the same IP-subnet, which is also the subnet of the advertised address.

The system is developed based on several assumptions. They have assumed that the local area network connecting the two servers as well as the Internet connection between the client and the server LAN will not suffer any permanent faults. The primary and backup hosts are
connected on the same IP-subnet. In FAWS primary backup should not be in the same IP-subnet.

In this system overhead is incorporated due to the use of proxies. Also the processing overhead increases linearly with the message size.

7. Performance Evaluation

In this section we discuss FAWS system performance evaluation based on various tests that were carried out. We performed experiments on Pentium IV (2.38 GHz) PC’s with 512MB RAM, running Windows XP professional, connected via 100Mbps Ethernet cards and a Netgear FS500 10/100MB fast Ethernet switch. The web servers were running Apache Tomcat 5.0.16. Each experiment consisted of ten apache axis clients accessing web services. We found out the amount of memory requirements of individual FAWS components on both environments as follows.

- FT-Front – 14 MB
- FT-Detector – 13 MB
- FT-Admin + FT-Monitor – 24 MB

![Figure 3: Failover time for service failure](image)

Figure 3 shows the graph of failover time distribution of the system when FT-Detector detects failure on web service this corresponds to the operation of FAWS described under the steps 8-13 on section 4. Calculated mean time to failover is 3.808s.

Failover time distribution for hardware failure on web server is shown in Figure 4 it corresponds to the operation described under steps 8-13 in section 4. Calculated mean value for the hardware failure failover time is 9.69s.

Failover time distribution when FT-Front gets an exception from web server is shown in Figure 5. This failover time is corresponds to the operation described under steps 14-15 in section 4. And calculated mean failover time is 3.61s.

![Figure 4: Failover time for hardware failure](image)

![Figure 5: Failover time for server exception.](image)

8. Future Work

FAWS was built with future extendibility and configurability in mind. Almost all its functionality can be redefined and extended. There is room for future enhancements. There are several features that can be added to make this solution more versatile.

**Active Replication** - The whole idea behind fault tolerance is to provide full availability of the system during web service failures and to prevent single point of failures in the system itself. Current system can provide system availability when there are failures in the web services. It is also possible to replicate each system components in the network such that it can tolerate single point of failures even in the system components.

**Load Balancing** - In the current system, there is only one primary server and it does the all the work in web services.
All the clients connect to this primary server through FT-Front. Backup servers remain in the system doing nothing. The system can be extended such that there are multiple primary servers in the FT-Group and share the workload. This can avoid congestion and server overloading when there are large numbers of web service requests coming to the system and they cannot be handled by a single server.

**Fail back operation** - The current FAWS system can be extended such that the service automatically fails back to the original primary server when a failed server comes back online. In the current system failed server must be removed from the FT-Group manually and should be added again when it recovers.

9. Conclusion

We have successfully conceptualized and implemented a client transparent fault-tolerant system for SOAP based web services. With the distributed object architecture and individual components, FAWS effectively increases the availability of web services in the presence of failures. Using both passive replication mechanism and message logging, it provides the level of reliability up to the client expectations. Also we were able to completely isolate the client applications from all fault tolerance mechanisms.

With the ability to work under any operating system environment, FAWS provides the portability and openness that cannot be seen in other such systems. Also the ability to failover without loosing any SOAP requests that are being processed at the time of failover is another feature of FAWS.

References