# Fault Tolerance in Web Services

Goutam Kumar Saha

gksaha@ieee.org

Web service applications can be made fault tolerant ones by means of affordable redundancy in data and function using Simple Object Access Protocol (SOAP) without using multiple versions or the N-Version fault tolerance model. Today fault tolerant web service application is of great importance. Web Service (e.g., weather report or latest traffic etc.) is a means to request information over the Internet through encoding of both the request and response in XML using standard Internet protocols for transport in order to make the messages universally available. Web Services are the major component of many modern Business-to-Business or B2B systems. Failures due to downtime or incorrect results in B2B systems might have considerable monetary penalties. Highly reliable B2B systems are must. Nowadays, WWW Consortium's (W3C) Simple Object Access Protocol (SOAP) has become the popular technology for developing web service application. SOAP is an Extensible Markup Language (XML) based protocol to let applications exchange information over Hyper Text Transfer Protocol (HTTP). SOAP is a protocol for accessing a Web Service. SOAP provides a way for two different systems to exchange information relating to a Remote Procedure Call (RPC) or other operation. It is important for application development to allow Internet communication between various programs. Conventionally, applications communicate using Remote Procedure Calls (RPC) between objects like Distributed Component Object Model (DCOM) and Common Object Request Broker Architecture (CORBA), but HTTP was not designed for this. RPC represents a compatibility and security problem, because firewalls and proxy servers normally block this kind of traffic. As all Internet browsers and servers support HTTP, it is better to communicate between various applications over HTTP, as a transport protocol. SOAP was created to accomplish this. SOAP provides a way to communicate between different applications running on different operating systems, with different technologies and programming languages. We need to use also Web Services Description Language (WSDL). WSDL is a document written in XML and it describes a Web service. WSDL provides a contract between a Web Service and the outside world. WSDL specifies the location of the service and the operations (or methods) the service exposes [*Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/ SOAP/]*. The Universal Discovery, Description and Integration protocol (UDDI) allows Web services to be registered so that they can be discovered by programmers and other Web services. We need to utilize both SOAP and WSDL because they are the most important Web services specifications, for better interoperability. In general, Web service is XML message that we send as the body of an HTTP POST request. The response is in XML that we then analyze. A typical Web Service is shown below [Goutam Kumar Saha*, Single Version Scheme of Fault Tolerant Web Services, ACM*

*Ubiquity, vol. 8(29), 2007*]. XML, SOAP [David Hunter, Andrew Watt, Jeff Rafter, Jon Duckett, Danny Ayers, Nicholas Chase, Joe Fawcett, Tom Gaven, Bill Patterson, *Beginning XML,* 3rd Ed. Wiley Publishing Inc. 2005] and Web Services give us a new paradigm for distributed computing [F. P. Coyle, *XML, Web Services and the Changing Face of Distributed Computing.* ACM Ubiquity, vol. 3(10), 2002], which includes XML as data, SOAP and HTTP as the protocols for moving data across the web; Web Services protocols like UDDI and WSDL for the discovery and connection those services. Web Services providers use UDDI to register a web service with the web services repository. Web Services client uses UDDI for discovering an appropriate web service and the way a client talks to a service provider depends on the WSDL.

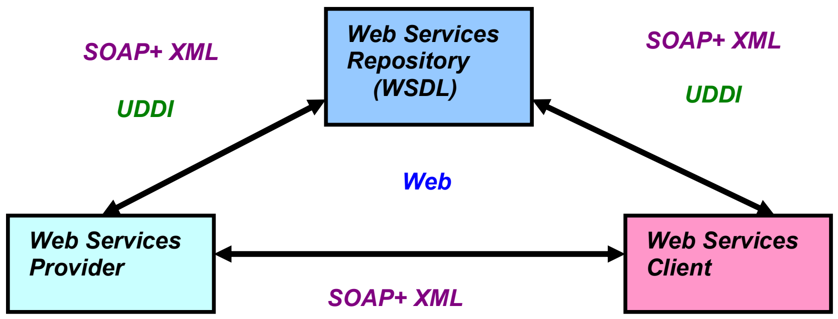

**Fig. Web Services**

The approach here is based on single version software (SVS) technique [Goutam Kumar Saha, *A Single- Version Scheme of Fault Tolerant Computing,* Journal of Computer Science & Technology, vol. 6(1), 2006] that relies on procedure triplication in order to tolerate one erroneous computation. This scheme detects errors by executing two copies of an application program with similar inputs and then comparing the results. Inequality in results indicates an error. Again, the approach is able to tolerate a fault or to mask errors through executing the three or more copies of an application program with similar inputs and then comparing or voting upon all the computing results for getting a result in majority. The Single Version Scheme of Fault Tolerance – the basic steps involved:-

***Step 0.*** Triplicate an application program in the form of a procedure: PC1, PC2, PC3

***Step 1**.* Sequentially execute: PC1, PC2 with similar inputs.

***Step 2.*** Validate the signature-based control-flow checking and then compare the outputs say, RC1 and RC2 of PC1 and PC2 respectively.

***Step 3.*** If both the outputs (values) are found to be same on comparison (no transient or permanent bit error has occurred or amidst *fail silent faults*), *then* application-system's output is RC1 or RC2.

***Step 4.*** If both the outputs (values) are not same, then execute the third image PC3 with similar input.

***Step 5.*** Validate the signature-based control-flows and compare the outputs (values) obtained from either of these application-copies that is, either RC1 and RC3, or RC2 and RC3, in order to find out equality and to output it.

***Step 6.*** If run-time signatures of control-flows are detected as erroneous, then compare (or vote) all the outputs from all the three replicas of an application, and if there is an output in majority, then application-system's output is the majority one only.

Here faults in either one of the application-replicas or bit-errors in run-time signatures are tolerated by masking the erroneous output (due to transient or permanent bit errors).

***Step 7.*** For no output in majority, the application is restarted or reloaded for re-execution. If a disagreement occurs, the SVS converges to a fail-stop kind scheme.

## SOAP IMPLEMENTED SVS –based FAULT TOLERANCE IN WEB SERVICES

In this work, we concentrate on a simple web based application, as an example, to exchange various process-parameters (for example, pressure, temperature values of a typical process) over Internet. Correct values need to be exchanged only. An odd or wrong response value needs to be masked out. In this fault tolerance approach, we have used three redundant XML documents (e.g., process_data_1, process_data_2 and process_data_3) using individual XML Schemas (that is, document model) [Goutam Kumar Saha, *Software Implemented Fault Tolerance – the ESVP Approach,* ACM Ubiquity, vol.7(31), August 2006]. We have also used replicated SOAP functions (e.g., GetProcessdata_1: to get data from process_data_1, GetProcessdata_2, GetProcessdata_3 and GetProcessdata_1Response etc). In WSDL, we have used three replicated operations. SOAP Requests (e.g., GetProcessdata_1, GetProcessdata_2 and GetProcessdata_3 are sent to a HTTP server) and three sets of Request- Response (on temperature and pressure for the same process) are received from the Server. An application then, needs to vote upon the received three values on temperature and to accept the major value as the correct value of temperature. For an example, the program code of a web service application, which is based on Single Version Scheme, can be viewed at [Goutam Kumar Saha, *Fault Tolerance in Web Services*, ACM Ubiquity, Vol.7(9), March 2006].We used code redundancy in designing fault tolerant web services. We get reliable web services on using three folded XML documents and SOAP functions. Such overhead in both time and memory space is affordable for a web application where we want to exchange sensitive data and thus to get a more reliable web services.