JOURNAL OF AERONAUTICS AND SPACE TECHNOLOGIES
JULY 2006 VOLUME 2 NUMBER 4 (23-28)

COMPARISON OF SOAP BASED TECHNOLOGIES: .NET REMOTING AND ASP.NET WEB SERVICES

Güray YILMAZ

Turkish Air Force Academy
Computer Engineering Dept.
Yeşilyurt/Istanbul
g.yilmaz@hho.edu.tr

ABSTRACT
Simple Object Access Protocol (SOAP) is a specification that enables applications to communicate with other applications [2]. Two major design goals for SOAP are simplicity and extensibility. SOAP attempts to meet these goals by omitting, from the messaging framework, features that are often found in distributed systems. In addition to this, SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment [1]. In this manner, SOAP provides reliable and robust message exchanging to such technologies, .NET Remoting and ASP.NET Web Services which are widely used.

Keywords: *Distributed Systems, SOAP, .NET Remoting, ASP.NET, Web Services.*

1. INTRODUCTION:

Simple Object Access Protocol (SOAP) is a specification that provides applications to communicate with other applications. Two major design goals for SOAP are simplicity and extensibility. SOAP attempts to meet these goals by omitting, from the messaging framework, features that are often found in distributed systems. Simple Object Access Protocol (SOAP) is a technology which is designed to achieve distribution of objects over the Internet. In fact, it is hard to provide distribution of objects and robust and reliable messaging between them due to Wide Area Network obstacles, such as scalability. SOAP which is proposed and developed by W3C (World Wide Web Consortium) 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 [1]. SOAP proposes the manipulation of a specific message format in Extensible Markup Language (XML) and a message transport protocol such as Hypertext Transfer Protocol (HTTP). Manipulating SOAP, different systems are able to communicate with each other by exchanging text messages encoded as XML. In fact, communication can be implemented over a transport protocol such as HTTP [2].

Fig. 1 and its explanation is taken from the article "SOAP: Simple Object Access Protocol" by William Bordes and Johann Dumser. In this figure the mechanism of SOAP is explained in details as a high-level diagram. The message is generated and translated to XML format and sent to other Application Server via HTTP. And other Application Server encodes the XML formatted message. The received message is controlled by XML Parser to check validity of the message using the HTTP and XML headers, it either rejects or accepts message. After validation, the requested application executes its task and the result is sent to caller application via same way.

SOAP message which enables application to communicate with each other has two main parts, including optional header and a required body. The header contains blocks of information relevant to how the message is to be processed. This includes routing and delivery settings, authentication or authorization assertions, and transaction contexts. The body contains the actual message to be delivered and processed. Anything that can be expressed in XML syntax can go in the body of a message. In Fig. 2, the parts of the SOAP message are showed in details.

SOAP has several major advantages to take place in the distributed systems. In fact, as mentioned above, SOAP is based on XML specification. In the nature of SOAP, it is a text-based message changing protocol. This provides processes to pass messages through

firewall without any security risks. It can be easily said that the content of SOAP message can not be encapsulate any hazardous data. In addition to this, SOAP is an open standard that is built upon open technologies such as XML and HTTP. In this manner, this specification provides de-facto standard for true distributed interoperability. On the other hand, SOAP has actual disadvantages that are based on its nature. As we mentioned above, SOAP is built upon HTTP protocol which requires a stateless request and response architecture. This means that SOAP specification does not offer permanent communication between participators.

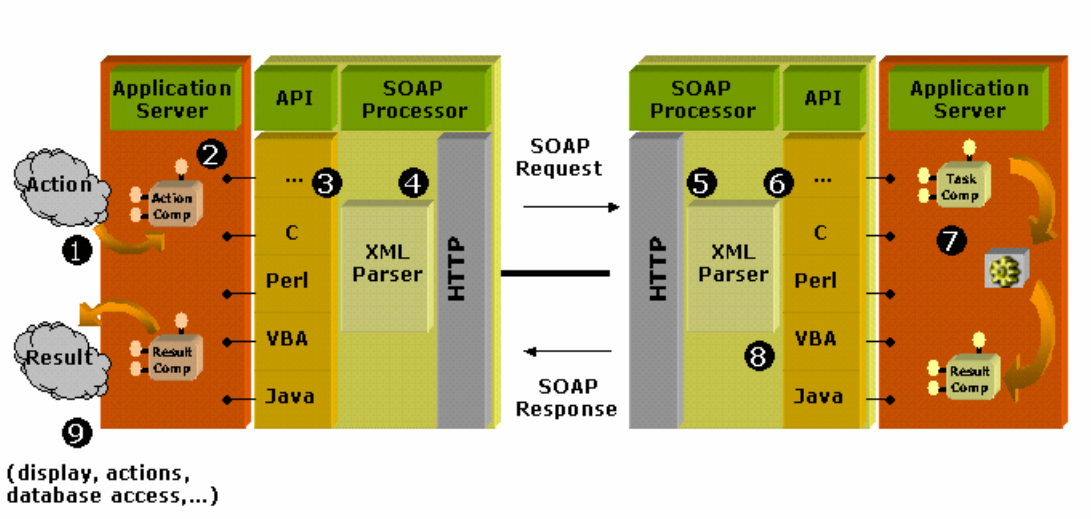

Figure 1. High-Level diagram of SOAP in a distributed system

Actually, SOAP is a specification rather than distributed architecture. For comparing SOAP with other distributed technologies, this difference is vital to indicate which distributed technologies are used and how the system is realized. For selecting distributed technologies to build application or general expression a system, there are actual some considerations, including scalability, performance, state management, garbage collection and security which are the foremost considerations. Table 1 shows how SOAP compares with the common distributed architectures based on these criteria [2].

Figure 2. The SOAP Message

2. .NET REMOTING FRAMEWORK

.NET Remoting Framework provides developer to build applications in which objects are distributed to different application domains to communicate with each other. Remoting in the .NET framework consists of numerous services that provide the ability to invoke objects that exist anywhere on the network. These objects could be in the same machine, on the same network, or located around the world. Only objects that are hosted within a CLR environment can be accessed using .NET Remoting.

In the essence of .NET Remoting, there are three major topics in which are discussed to imply .NET Remoting capabilities, including type of marshalling data, channel type and objects. According to type of marshalling data, two types of marshalling, binary formatter and SOAP formatter, are manipulated in .NET Remoting framework. Actually, these different formatter types are independent from the channel type. Another important key point in .NET Remoting is type of channel. The .NET Remoting layer supports pluggable channels how messages are sent. There are two standard channels for the message transfer, independent of format (i.e. Binary format or Soap format) both TCP Channel and HTTP Channel provides an implementation for a sender-receiver channel that uses the HTTP protocol to transmit messages [4].

Table 1. Comparison of SOAP with other distributed architectures

	CORBA	DCOM	JAVA-RMI	SOAP
Protocol Name	General Inter-ORB Protocol (GIOP)	Object Remote Procedure Call (ORPC)	JRMP	Any transport protocol.
Scalability	Corba uses stateful programming model which is not as scalable.	Least scalable. Clients ping the server at regular intervals to ascertain that it is still available. This pinging process limits scalability when large # of connections are involved.	Relatively scalable. Uses RMI Registry which could limit scalability if it is located on one server.	Most scalable of the four.
Performance	Once an object reference is obtained, CORBA permits direct client-server communication. Hence subsequent communication is very fast.	Requires several round-trips to activate and use the remote object. Once object's reference is obtained, direct object access without DCOM can take place from client.	Good performance. Works for Java language only and hence is fine-tuned for it.	Currently low. Overhead of extracting SOAP envelope, parsing XML, creating appropriate objects and converting parameters.
State Management	Connection-oriented and stateful.	Provides location transparency. Is stateful.	Very flexible. Provides both stateful and stateless sub-protocols.	Not addressed by SOAP. If HTTP is the protocol used, it is stateless.
Garbage Collection	CORBA does not address distributed memory management. Vendor-specific implementations exist.	Provides automatic garbage collection using the pinging mechanism discussed earlier.	Excellent garbage collection	SOAP does not address garbage collection.
Security	No intrinsic support for authentication, authorization or identity.	Very security-oriented. Provides support for authentication, authorization or identity. User can set appropriate level of security.	Since Java RMI works with java programming language it inherits the security built into Java. Use of RMI Security Manager can enable dynamic class loading thus providing additional security.	Since SOAP is a wire protocol, it does not address security. Security is determined by the transport protocol that it uses. For example, HTTPS using secured socket layer (SSL) when HTTP is the transport protocol.

The last characteristic and also called major topic of .NET Remoting is type of objects. There are three types of object which are offered from .NET Remoting Framework, including "Single Call", "Singleton Objects" and "Client-Activated Objects".

- Single Call objects are able to response one and only one request coming in. In this manner, Single Call objects are used in particular cases, e.g. where the objects are required to do a finite amount of work. Single Call objects are usually not required to store state information, and they cannot hold state

information between method calls. However, Single Call objects can be configured in a load-balanced fashion [3].

- Singleton objects are those objects that service multiple clients and hence share data by storing state information between client invocations. They are useful in cases in which data needs to be shared explicitly between clients and also in which the overhead of creating and maintaining objects is substantial [3].

- Client-activated objects (CAO) are server-side objects that are activated upon request from the client. This way of activating server objects is very similar to the classic COM coclass activation. When the client submits a request for a server object using "new" operator, an activation request message is sent to the remote application. The server then creates an instance of the requested class and returns an ObjRef back to the client application that invoked it. A proxy is then created on the client side using the ObjRef. The client's method calls will be executed on the proxy. Client-activated objects can store state information between method calls for its specific client and not across different client objects. Each invocation of "new" returns a proxy to an independent instance of the server type [3].

.NET Remoting Framework provides applications to invoke methods in another application domain. The other application domain or address space could be on the same machine or a different one. In this point, .NET Remoting Framework can be regarded as RPC in an object-oriented fashion. To sum up, overall mechanism of .NET Remoting (see Fig. 3) consists of:

1. Server Side Object is registered to one channel (its features are depended on the developer) and listened incoming messages from this channel.

2. Client Object marshals its invoking message with using proxy object to send remote server object.

3. The Remote server object gets the message and response it on the same channel.

3. ASP.NET WEB SERVICES

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine- processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [5]. In fact, as mentioned before, Web Services architecture is an open standard which was offered by W3C (World Wide Web Consortium). Web Services architecture allows programs written in different languages on different platforms to communicate with each other in a standards-based way. Also, it can be said that Web Services specification is not .NET Framework specific.

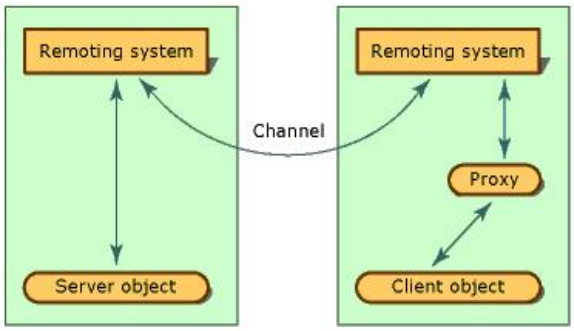

Figure 3. .NET Remoting framework general structure

Web Services expose useful functionality to users through a standard protocol. Mostly, the protocol manipulated for Web Services is SOAP. Web Services provide a way to describe their interfaces in enough detail to allow a user to build a client application to talk to them. This description is usually provided in an XML document called a Web Services Description Language (WSDL) document. Web Services are registered so that potential users can find them easily. This is done with Universal Discovery Description and Integration (UDDI). These UDDI services provide developers to integrate any operation which is offered by particular web service to their applications.

Web Services need five structures to realize remote communication between two applications, including discovery, description, message format, encoding and transport.

Discovery refers that it is necessary to resolve location of the remote service by the client application for which it requests to web service. The description of a web service contains structured metadata about the interface that is intended to be consumed by a client application as well as written documentation about the web service including examples of use. Message format, in order to exchange data, a client and a server have to agree on a common way to encode and format the messages. Encoding refers that the data transmitted between the client and the server needs to be encoded into the body of the message. Finally, transport refers that once the message has been formatted and the data has been serialized into the body of the message, the message must be transferred between the client and the server over some transport protocol [6].

Web Services mechanism is based on creating a service file which includes web methods for serving particular operations to its clients. In this mechanism, creating service and locating it are the major topics.

Due to the fact that the web service could work properly unless its location is lost by its client.

The ASP.NET Web Services structure offers a programming interface which is based on SOAP messages to method invocations. It accomplishes this by providing a very simple programming model based on mapping SOAP message exchanges to individual method invocations. The clients of ASP.NET Web Services do not have to know anything about the platform, object model, or programming language used to build them. The services themselves don't have to know anything about the clients that are sending them messages. The only requirement is that both parties agree on the format of the SOAP messages being produced and consumed, as defined by the Web service's contract definition expressed using WSDL and XML Schema (XSD).

ASP.NET Web Services rely on the System.Xml.Serialization.XmlSerializer class to marshal data to and from SOAP messages at runtime. For metadata, they generate WSDL and XSD definitions that describe what their messages contain. The reliance on pure WSDL and XSD makes ASP.NET Web Services metadata portable; it expresses data structures in a way that other Web service toolkits on different platforms and with different programming models can understand. In some cases, this imposes constraints on the types you can expose from a Web service—XmlSerializer will only marshal things that can be expressed in XSD. Specifically, XmlSerializer will not marshal object graphs and it has limited support for container types [7].

4. COMPARISON OF .NET REMOTING AND ASP.NET WEB SERVICES

The .NET Remoting and ASP.NET Web Services are both using SOAP specification, as mentioned above. It seems that both technologies can be regarded as in an RPC model. However, there are differences between these technologies according to their natures and usage fields.

First of all, ASP.NET based Web Services can only be accessed over HTTP. .NET Remoting can be used across any protocol. Actually, in .NET Remoting , the object can be registered to the channel with manipulating TCP or HTTP protocols. Thus, it can be easily seen that surplus information on communication channel is increasing when Web Services architecture is used. In other words, it causes to decrease the channel throughput. However, it is recommended that HTTP protocol provides robust communication when the communication environment takes place in World Area Network. Due to the fact that communication with using TCP may be obstructed by firewalls.

In addition to this, Web Services work in a stateless environment where each request results in a new object created to service the request. That's why ASP.NET Web Services are based on stateless communication. The client is recognized as new coming client whether it tries second or more communications or not. On the other hand, .NET Remoting supports state management options and can correlate multiple calls from the same client and support callbacks. Actually, as seen above, the singleton object model of .NET Remoting provides multi-user shared object state. This means the server of the object keeps track of the object state after the method invocations, and it could provide object state changing to its client.

Furthermore, Web Services serialize objects according to XML contained in the SOAP messages. For this reason, it can only handle with items that can be fully expressed in XML. .NET Remoting relies on the existence of the common language runtime assemblies that contain information about data types. In fact, this limits the information that must be passed about an object and allows objects to be passed by value or by reference. Web Services architecture provides simple programming structure, due to the fact that, it is based on XML specification. However, when attempting to realize more complex distributed application, this structure may not sufficient to perform both client and server operations as possible as.

Finally, Web Services support interoperability across platforms and are good for heterogeneous environments. .NET Remoting requires the clients be built using .NET, or another framework that supports .NET Remoting, which means a homogeneous environment. Thus, .NET Remoting is not flexible to build applications that are able to work on all platforms. In fact, it requires .NET Remoting Framework specific structures to enable communication. For instance, it is necessary to activate server side object from the client side when singleton object model is manipulated.

5. CONCLUSION

All things considered, when making decision for choosing .NET Remoting or ASP.NET Web Services, the important points are mentioned above. According to developer's point of view, Dhawan and Ewald, first, use ASP.NET Web services by default. They are simpler to implement and use, they offer the broadest possible reach to client platforms, and ASP.NET Web services client proxy code can be invoked from code run in a sandbox under the default security policy [7]. In fact, it is easy to learn and easy to implement Web Services due to its nature. In addition to this, although the .NET Remoting is proposed, developer should try to solve problem with using Web Services.

Actually, there is no chance to manipulate Web Services instead of .NET Remoting due to the characteristic of such problems. For instance, you have to arrange both client and server configuration with using objects simultaneously. This means that server side activated object is responsible for controlling the client side configuration as same as it changes its configuration. Web Services architecture does not allow system to perform this operation. In such example, .NET Remoting has to be chosen.

In conclusion, .NET Framework offers two SOAP based technologies, ASP.NET Web Services and .NET Remoting. .NET Remoting provides more complex structure, nevertheless, it offers more sophisticated operation comparing with the ASP.NET Web Services. On the other hand, ASP.NET Web Services provides more easier implementation structure comparing with the .NET Remoting Framework.

6. REFERENCES

[1] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, Dave Winer, *Simple Object Access Protocol (SOAP) 1.1.*, W3C Note 08 May 2000.

[2] Inder Nandrajog, *Simplified Object Access Protocol*, Management of IS, Spring, 2001.

[3] Paddy Srinivasan, *An Introduction to Microsoft .NET Remoting Framework*, Microsoft Corporation, July 2001.

[4] K.Sasikumar, *.NET Remoting,* c-sharpcorner. Com, February 2004.

[5] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard, *Web Services Architecture,* W3C Working Group Note, 11 February 2004.

[6] Scott Short, *Building XML Web Services For MS.NET Platform,* by Microsoft Corporation, 2002.

[7] Priya Dhawan, Tim Ewald, *ASP.NET Web Services or .NET Remoting: How to Choose*, Microsoft Developer Network, September 2002.

VITAE

Güray YILMAZ

He was graduated from Computer Engineering Department at The İstanbul Technical University, Istanbul in July 1991. He received his M.Sc. and PhD degrees in Computer Engineering from the same university in 1995 and 2002 respectively. He has been working an instructor at The Turkish Air Force Academy since 1991. His current research areas are operating systems, distributed systems, distributed object-oriented systems, web services.