SOAP - A SECURE AND RELIABLE CLIENT-SERVER COMMUNICATION FRAMEWORK

Marin Lungu, Dan – Ovidiu Andrei, Lucian - Florentin Barbulescu

University of Craiova, Faculty of Automation, Computers and Electronics,
Department of Computer Systems and Communication, Romania

Abstract: Data communication represents a key element within every client-server system as it provides the means for all the elements to connect and work together. This communication can be performed either by means of a proprietary protocol or by using a standard one, each option having its advantages and disadvantages. However the standardized protocols are more and more used in our days as it is easier to develop tools and applications. The current paper tries to compare some of the standard communication technologies with SOAP, the newest available on the market.

Keywords: networks, communication protocols, data transmissions

1. CLIENT - SERVER COMPUTING

Client/server computing is a phrase used to describe a model for computer networking. This model offers an efficient way to provide information and services to many users. A network connection is only made when information needs to be accessed by a user. This lack of a continuous network connection provides network efficiency. In client/server computing, processes are divided between the client and the server. This relationship is based on a series of requests and responses.

•Client: Requests services or information from another computer (the server computer).
•Server: Responds to the client's request by sending the results of the request back to the client computer.

In a client/server setting, the client computer runs a software application called a client program while the server computer runs a software application called a server program.

Functions performed by the client program:

•Enables the user to send a request for information to the server.
•Formats the request so that the server can understand it.
•Formats the response from the server in a way that the user can read.

Functions performed by the server program:
•Receives a request from a client and processes the request.
•Responds by sending the requested information back to the client.

The following diagram illustrates the relationship between client and server computers. The client requests information; the server processes the request and sends a response back to the client

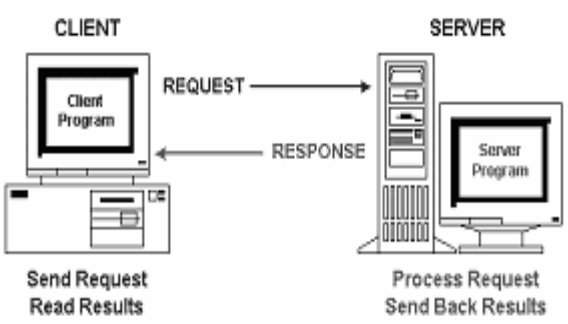

Fig. 1. Standard Client-server arhitecture. The client requests information; the server processes the request and sends a response back to the client.

In conclusion, client/server computing is a common networking model which enables many users to access information in an efficient manner. Generally, the user's computer is called the client and the machine that contains the information being accessed is called the server. The client computer runs an application called a client program. A client program enables a user to send a request for information to the server and read the results that the server sends back. The server computer runs a server program which processes requests and sends results back to the client. The communication between the client and the server is based on request-response protocols

2. DISTRIBUTED TECHNOLOGIES

There are several request/response based communications technologies on the market today. It is impossible to say that one of them is the best or the worse, but is easy to highlight their strong and weak points. The final decision in choosing a communication techonology must be made based on the specifications of the client-server system that must be implemented. In this paper we wil cover only five of those technologies: CORBA, DCOM, Java/RMI, XML-RPC and SOAP.

2.1 CORBA

Common Object Request Broker Architecture relies on a protocol called the Internet Inter-ORB Protocol (IIOP) for remoting objects. Everything in the CORBA architecture depends on an Object Request Broker (ORB). The ORB acts as a central Object Bus over which each CORBA object interacts transparently with other CORBA objects located either locally or remotely. Each CORBA server object has an interface and exposes a set of methods. To request a service, a CORBA client acquires an object reference to a CORBA server object (Gopalan 1998).

The client can now make method calls on the object reference as if the CORBA server object resided in the client's address space. The ORB is responsible for finding a CORBA object's implementation, preparing it to receive requests, communicate requests to it and carry the reply back to the client. A CORBA object interacts with the ORB either through the ORB interface or through an Object Adapter - either a Basic Object Adapter (BOA) or a Portable Object Adapter (POA).

Since CORBA is just a specification, it can be used on diverse operating system platforms from mainframes to UNIX boxes to Windows machines to handheld devices as long as there is an ORB implementation for that platform. Major ORB vendors like Inprise have CORBA ORB implementations through their VisiBroker product for Windows, UNIX and mainframe platforms and Iona through their Orbix product. (Gopalan 1998)

2.2 DCOM

Distributed Component Object Model which is often called 'COM on the wire', supports remoting objects by running on a protocol called the Object Remote Procedure Call (ORPC). This ORPC layer is built on top of DCE's RPC and interacts with COM's run-time services. A DCOM server is a body of code that is capable of serving up objects of a particular type at runtime. Each DCOM server object can support multiple interfaces each representing a different behavior of the object.

A DCOM client calls into the exposed methods of a DCOM server by acquiring a pointer to one of the server object's interfaces. The client object then starts calling the server object's exposed methods through the acquired interface pointer as if the server object resided in the client's address space. As specified by COM, a server object's memory layout conforms to the C++ vtable layout. Since the COM specification is at the binary level it allows DCOM server components to be written in diverse programming languages like C++, Java, Object Pascal (Delphi), Visual Basic and even COBOL. As long as a platform supports COM services, DCOM can be used on that platform.

DCOM is now heavily used on the Windows platform. Companies like Software AG provide COM service implementations through their EntireX product for UNIX, Linux and mainframe platforms; Digital for the Open VMS platform and Microsoft for Windows and Solaris platforms. (Gopalan 1998)

2.3 Java/RMI

Remote Method Invocation relies on a protocol called the Java Remote Method Protocol (JRMP). Java relies heavily on Java Object Serialization, which allows objects to be marshaled (or transmitted) as a stream. Since Java Object Serialization is specific to Java, both the Java/RMI server object and the client object have to be written in Java.

Each Java/RMI Server object defines an interface which can be used to access the server object outside of the current Java Virtual Machine(JVM) and on another machine's JVM. The interface exposes a set of methods which are indicative of the services offered by the server object. For a client to locate a server object for the first time, RMI depends on a naming mechanism called an RMIRegistry that runs on the Server machine and holds information about available Server Objects.

A Java/RMI client acquires an object reference to a Java/RMI server object by doing a lookup for a Server Object reference and invokes methods on the Server Object as if the Java/RMI server object resided in the client's address space. Java/RMI server objects are named using URLs and for a client to acquire a server object reference, it should specify the URL of the server object as you would with the URL to a HTML page. Since Java/RMI relies on Java, it can be used on diverse operating system platforms from mainframes to UNIX boxes to Windows machines to handheld devices as long as there is a Java Virtual Machine (JVM) implementation for that platform. In addition to Javasoft and Microsoft, a lot of other companies have announced Java Virtual Machine ports. (Gopalan 1998)

2.4 XML-RPC

EXtensible Markup Language - Remote Procedure Call was created in 1998 by David Winer. Winer felt that the ones already in use (DCOM, CORBA) were not that suitable for the Internet. XML-RPC is a specification and a set of implementations that allows computers to communicate with each other and make procedure calls over the Internet.

It works by encoding the RPC requests into XML and then sending them over a standard HTTP connection to the server.The server then decodes the XML, executes the function and then sends the result back to the client in XML. The client decodes the XML and carries on executing as before. XML-RPC is designed to be simple, it is easy to use, understand and debug (as the RPC protocol is in XML this makes network sessions are easier to debug). There are XML-RPC implementations for most popular programming languages and environments. XML-RPC is basically a "remote procedure calling using HTTP as the transport and XML as the encoding. (Johnson 2001)

2.5 SOAP

Simple Object Access Protocol is an extension to the XML-RPC. There are two different ways of implementing RPCs which both rely on XML and HTTP for their implementation. SOAP picks up from where XML-RPC left off by implementing user-defined data types and including the ability to specify the recipient, message specific processing control among other features.

XML-RPC consists of simple, easy to understand requests and responses and allows a job to get done with the minimum amount of complexity. SOAP, on the other hand, requires attribute specification tags, namespaces and other complexities, which mean that there is an increase in the amount of overheads. However, there is more information about the messages being sent. In brief, SOAP is used whenever complex user-defined data types are used and it is required to specify how the message is to be processed; otherwise for simple method calls and standard data types XML-RPC can be used. (Rivera 2001)

3. TECHNOLOGIES COMPARISON

SOAP as described earlier on, is a wire protocol. Therefore comparisons between CORBA and RMI are not possible - they are architectures set up for distribution technology. IIOP and JRMP are the wire protocols for the two architectures. These can therefore be compared to SOAP. DCOM's protocol ORPC could also be compared but future development for this architecture has stopped due to Microsoft's adoption of SOAP. Having said that, adoption of IIOP constricts you to using CORBA as the architecture while JRMP constricts you to RMI. Therefore in some aspects of comparison it is possible to compare SOAP with CORBA and RMI.

It has been said that SOAP doesn't address high level object functions such as object activation, lifetime management, polymorphism, bi-directional communication or garbage collection. However neither do JRMP or IIOP. The point of the wire protocol is to transmit the object information in a standard way. CORBA and RMI have layers above the JRMP and IOP which do these high level object functions. Therefore this implementation has been left open to the developer to create or to use a vendor based product which will include this functionality with SOAP (provide a framework.)

It has also been said that SOAP is also stateless if it binds to HTTP, it does not rely on previous requests. Again we are talking about the higher layers doing this work. Currently SOAP implementations do not have a session mechanism to enable transactional requests. Therefore huge amounts of data may need to be continually transmitted. However steps are going ahead to remedy this problem as even HTTP has cookies to enable stateful transactions. CORBA, RMI and DCOM all support stateful requests with RMI giving the option to create stateless requests

3.1 Infrastructure Comparisons.

Heterogeneous Operating Systems/Language Environment. In RMI/JRMP's case Java Object Serialisation is only specific to Java, which means that both the client and the component object must be written in Java. CORBA/IIOP can also function with different hardware and software as long as the ORBs are the same on the different machines. What is encoded then decoded is interpreted as the same at both end points. SOAP can function with different

hardware and software however it needs the higher level interpretation of the XML to be the same. This is paralleled with the ORBs in CORBA's case. However in SOAP's case this can be avoided if standard types are used. The mappings of these primitive types are the same as it is in the specification, if not then the libraries of the SOAP implementation must be the same or compatible

Business Considerations: SOAP with HTTP transport binding means that companies are familiar with using HTTP. The business infrastructure built to integrate the web and HTTP into the working place can now be reused. CORBA and RMI require extra infrastructure to be implemented.

Microsoft and IBM support SOAP. They are the major players of the software developers which means that other companies will adopt it. Therefore it is an issue on whether interoperability with other companies may also need to be addressed.

Companies also have to take into account the cost involved in adopting certain distributed technologies e.g. CORBA requires a licence whilst RMI is free. With SOAP, there are free open source implementations or pre-packaged software from vendors. However, with the pre-packaged software true interoperability between products from different vendors may not be achieved due to proprietary features.

Firewall filtering. The issue of firewalls has meant that the use of RMI, CORBA has been restricted over the Internet. Although ports can be set up for transmission, dynamically changing these ports means recompiling the program or a scope of ports must be made open by the firewall. This requires the firewalls at both ends of communication to be configured in exactly the same way. There are however some IIOP friendly firewalls or some firewalls which do IIOP-HTTP tunnelling.

SOAP does not need such major changes to the firewall or the network. It can run over the same ports as web applications. SOAP messages call their intent inside the HTTP header, so it is possible for firewalls to filter based on this information. SOAP server's responsibility must check the HTTP header with that in the headers and tags in the XML payload otherwise it is rejected. Firewalls can easily recognise SOAP packets based on their Content Type and can filter based on the interface and method name exposed via the HTTP headers. IIOP and JRMP are encoded as bit streams so it is difficult to decode to determine their intent.

The SOAP specification states that additional HTTP headers must be introduced, which makes sure that certain headers are recognised and understood before processing the request. This is done using M-POST requests. The specification requires first to implement the request using MPOST then if it gets the response "501 not implemented" or "510 Not extended" it could try again using normal POST. Therefore SOAP clients send this in, only when it fails use normal POST

Integration with other distributed technologies. The main problem of the distributed technologies is that they are incompatible with each other e.g. a DCOM based system cannot talk to an RMI based system. Sun have tried to tackle that problem and RMI is now compatible with both IIOP and JRMP.

SOAP can be used as the lower common layer for the architecture. A certain communication language can be encoded into SOAP and decoded into appropriate communication language at the receiver end. Therefore it can help different technologies to communicate without having too mush of an impact on the architecture already set up on the systems.

3.2 Functional Comparisons

Serialisation of Objects. SOAP can support serialisable objects by converting it into a XML element. It has the same functionality as IIOP and JRMP in this sense. Toolkits are available to hide the SOAP implementation using keywords in the language and can be used to convert certain function calls from a certain language into SOAP and parse it back. The type of function calls, datatypes of the parameters supported vary with each SOAP implementation. Therefore one which uses HTTP and another using SMTP would both conform to the SOAP standard but would not be able to communicate with each other. Complying with the standard does not mean that they will all interoperate even though this is a goal of some toolkit makers

Performance. SOAP currently has a lack of performance due to the requirement to create, parse and transport XML. Even more so when using unicode. It uses a lot of memory compared to the amount of actual data in the document and is a time consuming task. Transmitting a lot of data may be unsatisfactory in a high throughput situation and also will produce a high overhead as the actual information is only a small portion of the total data.

IIOP is very fast because it permits direct client server communication once an object reference is obtained. JRMP's performance has been refined since it caters for only one language. Both of these protocols are sent as bit streams as opposed to text and are less descriptive. Therefore the decoding of the bit stream will always be faster.

Programming Usability. Given that the IT community has been exposed extensively to XML and HTTP compared to the few with specialized knowledge with using CORBA, RMI and DCOM,

more people have the basic knowledge needed already to implement and understand SOAP.

With RMI and CORBA, there is compile time type checking of the parameters of the function available, however there is no such feature with SOAP; there will not be a type error until the XML message is validated on the server at runtime. The nature in which it is binded to HTTP or other protocols means that there is an independence from the client to the server, it is loosely coupled. Unlike JRMP and IIOP, using SOAP means that the server can be replaced with a new one without the client knowing about it. However, more runtime checking and asynchronous communication is needed. Runtime exceptions in RMI are wrapped in remote exception therefore some knowledge in the calling application of an RMI object is required for diagnosis of errors. This means there is a lack of information on what went wrong.

SOAP requires more code than other distributed technologies but it is also human readable if intercepted unlike IIOP and JRMP which is sent as serialised data and would just be incomprehensible. Therefore SOAP messages are easier to debug for the programmer. Whether readable text is of great benefit, when it will encrypted over the Internet anyway means that this advantage is lost.

Security. SOAP security has yet to be fully addressed. It was thought that HTTP security would just be used. HTTP provides several ways to authenticate which user is making the SOAP call, but does not when it needs to be propagated from different transports (like HTTP and SMTP). There is also a need to secure the entire SOAP message including the SOAP headers and the SOAP body for transmission over the Internet. This is done using XML security tags. A security layer needs to be added on top of SOAP for this to be implemented. IBM and Microsoft have proposed an open security proposal: to extend SOAP with attachments, security extension and digital signatures thus adding security to protocol level.

IIOP leaves this implementation to be done via the ORB and JRMP also leaves this to be done by its interface layer in RMI. RMI has built in security features already due to Java.

4. SOAP – THE BEST CHOICE IN MOST CASES

As it was presented earlier in this paper choosing the best communication framework is strictly related to the needs of the implemented client-server system. Usualy the choosen technology must be secure, reliable, fast and possibly platform independent. From the technologies presented SOAP comply to all of those four requirements.

The security issue was a problem for SOAP. Initialy this protocol wasn't designed to contain built-in security. When SOAP was first created the security issue was left to be implemented by the transport protocol. That is why SOAP can be used over HTTPS and thus the security of this protocol will be used by SOAP. But this can only ensure that during the transport the informations ent can not be altered and not that the request or response come from the correct entity. It was cleared that some kind of security must be added to the protocol itself. This is where the fact that SOAP is based on XML come to the rescue. XML Signature and XML encryption are two innovations added to XML.. By exchanging SOAP messages that contain signed XML bodies the client and the server can check if the data transfered between them come indead from the right source and was not altered. By encrypting the XML the data can only be read the be intended target leaving no room for message interception by unauthorized entities.

About the reliability of SOAP there is nothing that can be said. SOAP is based on XML. XML is known for it's reliability. Incomplete data will be detected immediately because of the strong typed structure of XML. Also, the data corruption can be signaled in this way. There can still be a problem related to data corruption: The XML parser can not detect corruption in the data between two XML tags. This is not the case when XML signature is used.

The speed of transfer is indeed a problem for SOAP especialy when binary data neads to be transfered. Usualy this data is serialized into a plain text representation (usualy BASE 64) which leads to messages about 50% bigger that the corresponding binary data streams. The solution found here was to store the binary files on servers and to use only references to them in exchanged SOAP messages. XML explicitly supports referencing external opaque data as external unparsed general entities. Considered a fairly esoteric feature of XML, unparsed entities are not widely used. The primary obstacle to using unparsed entities is their heavy reliance on DTDs, which impedes modularity as well as use of XML namespaces. They are also not available to SOAP, which explicitly prohibits document type declarations in messages. A more common way to reference external opaque data is to simply use a URI as an element or attribute value. (Bosworth *et al.* 2003)

The last issue, the platform independence, is passed very easy by SOAP because of the platform independence of the two technologies it uses (XML for data representation and HTTP for network transport). In the last years SOAP is more and more use especialy when a communication between different types of systems is neaded.

In conclusion SOAP offers the most advantages over other similar communication protocols. It’s easily understandable structure, its platform independence

and the huge support it has from some of the major actors of the computer market makes SOAP the best choice when a communication framework is neaded.

REFERENCES

Bosworth, A, D. Box, M. Gudgin,M. Nottingham, D. Orchard and J. Schlimmer (2003): *XML, SOAP, and Binary Data*.

Gopalan, S. R. (1998). *A Detailed Comparison of CORBA, DCOM and Java/RMI*

Johnson, J (2001): *Using XML-RPC for Web services.*

Network Solutiuon CO. (1996): *What is client/server computing?.*

Rivera, G (2001*): Some considerations on SOAP*