Adaptive QoS for Mobile Web Services through Cross-Layer Communication

P.Sowjanya
Asst.Prof
Sowjanya.ponnuru@gmail.com

ABSTRACT
As Web services become increasingly popular, more businesses are building their future solutions around Web service technology. For service providers, quality of service is a key aspect of this technology. Although service offers with no QoS guarantees are acceptable in some simple scenarios, QoS is critical in selecting Web services as building blocks in more sophisticated business applications. For this reason, standards such as the Business Process Execution Language for Web Services (BPEL4WS) incorporate security, reliability, accessibility, and other QoS features. Two demands are driving the need for QoS specifications in Web services. On one hand, clients seek reliable service performance whenever the need arises. On the other hand, service providers strive to achieve an optimal balance between user satisfaction and system utilization; for example, crucial transactions such as payment should be executed at high priority. Given the ubiquity of mobile devices, clients using them will likely generate a large percentage of all Web service requests in the future. However, mobile devices are resource-constrained in terms of CPU, memory, and battery life. This is problematic because, compared to traditional Web interaction, Web services impose additional overhead. For example, SOAP requests and replies are four to five times larger if implemented as a Web service. Parsing XML code also adds extra computing costs.

Compressing Web service interactions is therefore desirable. Although compression and decompression require more computing power, compression is still useful in many cases because decompression causes an almost negligible performance loss and wireless transmission consumes far more energy than CPU operations. Further, service providers charge mobile clients by volume rather than by connection time.

KEYWORDS
WSBL, UDDI, EAI, EII.

INTRODUCTION
As Web services become increasingly popular, more businesses are building their future solutions around Web service technology. For service providers, quality of service is a key aspect of this technology. Although service offers with no QoS guarantees are acceptable in some simple scenarios, QoS is critical in selecting Web services as building blocks in more sophisticated business applications. For this reason, standards such as the Business Process Execution Language for Web Services (BPEL4WS) incorporate security, reliability, accessibility, and other QoS features. Two demands are driving the need for QoS specifications in Web services. On one hand, clients seek reliable service performance whenever the need arises. On the other hand, service providers strive to achieve an optimal balance between user satisfaction and system utilization; for example, crucial transactions such as payment should be executed at high priority. Given the ubiquity of mobile devices, clients using them will likely generate a large percentage of all Web service requests in the future. However, mobile devices are resource-constrained in terms of CPU, memory, and battery life. This is problematic because, compared to traditional Web interaction, Web services impose additional overhead. For example, SOAP requests and replies are four to five times larger if implemented as a Web service. Parsing XML code also adds extra computing costs.
Compressing Web service interactions is therefore desirable. Although compression and decompression require more computing power, compression is still useful in many cases because decompression causes an almost negligible performance loss and wireless transmission consumes far more energy than CPU operations. Further, service providers charge mobile clients by volume rather than by connection time.

About Java

Initially the language was called as “oak” but it was renamed as “Java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

1. Java is a programmer’s language.
2. Java is cohesive and consistent.
3. Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.
4. Finally, Java is to Internet programming where C was to system programming.

Importance of Java to the Internet

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

Java can be used to create two types of programs

Applications and Applets: An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java’s ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

Features Of Java Security

Every time you that you download a “normal” program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scanned them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a “firewall” between a network application and your computer.

When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.
Portability

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed. As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java’s solution to these two problems is both elegant and efficient.

The Byte code

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

Java Virtual Machine (JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that’s has been generated by the compiler will not corrupt the machine that it’s loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

Overall Description

Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a. class file, which contains the byte code. The Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of code

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the
issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

**Compiling and interpreting Java Source Code**

During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be an Intel Pentium Windows 95 or SunSARC station running Solaris or Apple Macintosh running system and all could receive code from any computer through Internet and run the Applets.

**Simple**

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ programmer, learning Java will be even easier. Because Java inherits the C/C++ syntax and many of the object oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

**Object-Oriented**

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

**Robust**

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time.

Java virtually eliminates the problems of memory management and deallocation, which is completely automatic. In a well-written Java program, all run time errors can –and should–be managed by your program.

**1.2 SCOPE:**

This application has a very wide area of usage. As the number of increases, the scope of project goes on. There is no limit for this application. With this type of application, the knowledge can be distributed to a large extent.

**1.2.1 Scope of the system**

The scope of the system includes:

1. Handling the both service providers and service clients requests.
2. Enables efficient service offer discovery and selection to accelerate the overall lookup process for clients;
3. Provides a flexible way for service providers to publish and update service offers with different QoS aspects;

**1.2.2. Features.**

The main objective of the system is to develop a Qos service by using the predefined parameters which specify the users requirements in Qos offers. Some times it may need to
facilitate the user to define his one parameters to specify the requirements. By considering the layers in the web service communication we can achieve the maximum utilization of the resources.

**SUBTITLES**

1. UDDI Registry system
2. Offer Broker Module.
3. Service Provider Module.
4. Client Application Module

**CONCLUSION**

In the present system we are developing the QoS-service which can use the cross layer communication architecture and as well as using the web service broker to look up the service client requirements to discover the service providers. And it updates the service providers offers which provides the updated information.

By using the WSB it reduces the nine steps involved in communication are reduced to 4 steps in clients point of view.

**FUTURE SCOPE**

We are trying to increase the QoS-service, and to reduce the no. of steps from 4.

**REFERENCES**

[1] Java Complete Reference - Herbert Shield


