A Comparison of Java RMI, CORBA, and Web Services Technologies for Distributed SIP Applications

Mark D. Hanes Stanley C. Ahalt Ashok K. Krishnamurthy

Department of Electrical Engineering
The Ohio State University

Abstract

An emerging trend in the Signal and Image Processing (SIP) community is the appearance of middleware and middleware standards that can be readily exploited for distributed computing applications by the SIP community. High performance computing and High Performance Embedded Computing (HPEC) applications will benefit significantly from highly efficient & portable computational middleware for signal & image processing. Open middleware standards such as VSIPL, MPI, CORBA, Java RMI, and Web Services (based on SOAP/XML), offer a unique opportunity for the rapid development of easily maintained HPEC codes that combine portability and flexibility across a number of applications This middleware infrastructure will support the rapid development and deployment of portable, efficient, SIP-critical applications that will be of immediate benefit to many.

The use of distributed computing technologies for problem solving has been around for many years. The early paradigm of distributed computing has been that of remote procedure calls (RPC). However, in recent years, this paradigm has shifted to the use of remote objects due to the acceptance of object oriented programming practices. Even today web services are built around the concept of messaging and frequently these messages take the form of request/response-type remote procedure calls on remote objects. The existing and emerging standards for performing distributed computing have resulted in several possible middleware choices for the SIP community.

This paper focuses on three specific middleware standards for distributed computing, namely: the Common Object Request Broker Architecture (CORBA), Java's Remote Method Invocation (RMI), and the more recent web services technology, frequently based on XML data encoding and SOAP-based messaging protocols. More specifically, we will be focused on the appropriate use of such technologies for implementing new SIP applications, or extending legacy applications through the use of these technologies.

The three middleware standards we have selected all have certain commonalities. All are based around the concept of a client application using the services available on a remote machine, or server. A remote executable object that implements one or more exposed interfaces provides these services. The object's interface represents a contract between the client and the server. This interface is written as a Java interface for Java RMI, in IDL for CORBA, and in WSDL for web services. In the latter two cases, the more generic descriptions can be translated into specific language implementations, although a standard translation of WSDL into a wide variety of languages is still being developed.

All three technologies also contain the notion that a client application need not know the exact network location of an object prior to runtime. A 'discovery' process exists (with varying levels of sophistication)

to enable the client to obtain a handle to an object that implements a particular desired interface. This discovery takes the form of a Naming Registry or Service in Java RMI and CORBA, and WSDL repositories or UDDI in Web Services.

Each middleware standard also has certain differences that are unique to the technology. Java RMI, first released around 1997, provides a distributed computing platform specifically focused on Java-based clients and servers. Due to Java's inherent platform-independent capabilities, RMI-based applications are capable of running on a wide variety of computing platforms. This represents both a strength and weakness of Java RMI, however. The weakness is due to RMI's heavy reliance on Java and lack of direct support for other common languages, such as C or C++.

The first Java RMI wire protocol, JRMP (Java Remote Method Protocol) provided a mechanism for passing both primitive types and objects across the network. Classes (executable code) can actually be loaded dynamically across a network through the appropriate use of security managers and dynamic class loaders. Initially, JRMP was unable to effectively interoperate with other distributed computing applications that were based on CORBA technologies. However, in 1999, the J2EE platform integrated JRMP with CORBA's IIOP (Internet Inter-Orb Protocol), thus providing drastically improved interoperability.

CORBA, whose specification was first released in the early 1990's, chose a new interface description language, known as the IDL, for defining its object interfaces. This intermediate language can then be used to translate to/from a variety of different languages, such as C, C++, and Java. This makes CORBA more suited toward integration with legacy systems written in languages created before Java, in that it is language agnostic.

The concept of a web service first appeared in the late 1990's and early 2000's. The exact definition of a web service is still in flux, but loosely speaking it can be viewed as a collection of methods (with hidden implementations) that can be discovered and invoked by a client application. The network wire protocols used in Web Services are typically XML-based and ride on a network protocol such as HTTP, HTTPS, SMTP, *et al*. A protocol known as SOAP is currently in use for describing the messages sent to/from web services and is itself an application of XML. Web services are language agnostic, much like CORBA, in that the interfaces are described in WSDL, which, like IDL, is independent of any one particular computing language.

This paper focuses on the use of these three middleware standards for performing distributed computing as applied to signal and image processing (SIP) applications. We explore in greater depth the important tradeoffs in selection of these technologies for SIP applications. We demonstrate the use of the technologies for a specific SIP application, focusing on the use of the technologies for data retrieval, computation, and interfacing to legacy code, written in computing or even matrix-manipulation languages.

References:

Brose, Noffke, and Muller, *JacORB 1.3 Programming Guide*, www.jacorb.org, 2001.
Brose, Vogel, and Duddy, *Java Programming with CORBA*, Wiley Computer Publishing, 2001.
Graham et al., *Building Web Services with Java*, Sams Publishing, 2002.
Seshadri, *Enterprise Java Computing: Applications and Architecture*, Cambridge University Press, 1999.
Snell, Tidwell, and Kulchenko, *Programming Web Services with SOAP*, O'Reilly, 2002.