International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Analysis of Distributed Object Technologies

Dr. Sanjay E. Yedey

P. G. Department of Computer Science and Technology, DCPE, Amravati, Maharashtra, India

Abstract: ***The advancement in hardware as well as software technologies leads towards more and more use of computers in day to day life. Every field from Education, Business, Agriculture, Sports, Entertainment or whatever, the use of computers became an essential part. All the fields need data, resources as well as applications required to be shared among the users which leads towards the development of distributed systems. The most popular and challenging among distributed technologies is 'Distributed Objects'. The Distributed Object technology allows objects active in one process be accessed by another facilitating the computation be split over multiple processes. The processes involved may be running in different address spaces on single system or may be on different systems in a network in a local area network or the Internet in different networks may be having the systems located seven sees apart. The most popular distributed object technologies are CORBA, RMI and DCOM. This paper presents an analysis of architecture and working of these technologies and presents performance comparison on the basis of data marshalling, interoperability, heterogeneity, design transparency and speed.***

Keywords: Distributed Objects, CORBA, RMI, Marshalling, Interoperability, Heterogeneity, Design Transparency

1. Introduction

With the advancement of the technology and the growth of the Internet, the use of Internet has been increased many folds. Many applications useful in our day to day life are available on the internet and are accessed at ease using portable devices like smart phones and laptops along with micro to mainframe or even supercomputers. This creates an environment of heterogeneous distributed systems spread across all over the world. The requests from large number of users create a tremendous burden on the server, affecting performance of the services on web server. The high performance is required in many server situations such as Hyper Text Transfer Protocol (HTTP) [1] and File Transfer Protocol (FTP) [2], e-mail[5], chat[6], etc. Although a centralized system approach facilitated by powerful systems like supercomputer or mainframe computer was employed to solve the problem at an early stage, it had its own limitations with respect to aspects like bandwidth, traffic congestion, reliability and even cost. The limitations made client/server [3] and distributed processing[3][4] approach more suitable. In addition, the fast growth of a network performance [7] has accelerated the multiple computer approaches. A distributed processing is widely used these days, especially in multi-tier environments.

Among the distributed processing technologies, Remote Procedure Call (RPC) [10] and Remote Method Invocation (RMI) [11] exist as early models. However, as the object oriented paradigms flourished, these models have been evolved into distributed object technologies such as Distributed Component Object Model (DCOM) [12], Common Object Request Broker Architecture (CORBA) [13], and along with RMI.

This paper describes the architecture and working of these technologies along with advantages and disadvantages of each technology, providing a guideline to the developers, vendors, and practitioners to help them to choose an appropriate technology to develop mission critical application in distributed environment.

2. Distributed Systems

A distributed system consists of a collection of autonomous computers linked by a computer network equipped with distributed system software. This software enables computers to coordinate their activities and to share the resources of the system hardware, software and data. Users of a distributed system should perceive a single, integrated computing facility even though it may be implemented by many computers in different locations. This is in contrast to a network, where the user is aware that there are several machines whose locations, storage replications, load balancing and functionality are not transparent. Benefits of distributed systems include bridging geographic distances, improving performance and availability, maintaining autonomy, reducing cost and allowing for interaction.

3. Distributed Objects

In distributed system technologies, the concept of '**Distributed Objects'** refers to a technique in which 'objects' are distributed across different address spaces, either in different processes on the same computer, or even in multiple computers connected via a network. Distributed object models and tools extend an object-oriented programming system. These objects, though active on different machines, work together in collaboration by sharing data and invoking methods. This communication often involves location transparency, where remotely located objects appear the same as local objects. The principal way of communication among these distributed objects is by using 'Remote Method Invocation (RMI)', generally by message-passing. In message-passing, one object sends a message to another object in a remote machine or process to perform some task. The results are sent back to the calling object.

The objects may be distributed on different computers throughout a network, living within their own dynamic library outside of an application, and yet appear as though they were local within the application. This is the essence of plug-and-play software. Several technical advantages result from a distributed object environment. The overall technical

goal of distributed object computing is to advance distributed information technologies so that they may be more efficient and flexible, yet less complex. The benefits of distributed objects are indeed solutions to the problems with existing, monolithic client/server paradigms. [3]

3.1 CORBA

CORBA is part of the *Object Management Architecture* (OMA), developed by OMG, which is also the broadest distributed object middleware available in terms of scope. It allows integration of a wide variety of object systems. The basic OMA reference model from the OMG specification presents CORBA architecture [11]. The *Object Request Broker* (ORB) component enables clients and objects to communicate in a distributed environment. Four categories of object interfaces use ORB to interact:

- *Object Services* are interfaces for general services that are likely to be used in any program based on distributed objects.
- *Common Facilities* are interfaces for horizontal end-user-oriented facilities applicable to most application domains.
- *Domain Interfaces* are application domain-specific interfaces, which may also be a collection of different Domain Interfaces such as Finance, Telecom, Transportation, etc.
- *Application Interfaces* are non-standardized application-specific interfaces.

The key component in OMA is ORB, or specified as CORBA. From the above description, it is not hard to see that ORB needs to provide the functions of delivering requests to objects and returning any responses to the clients targeted. As a distributed environment, ORB shall also support the transparency requirement. CORBA presents a nice architecture of an ORB, which handles a series of jobs like object allocation, object implementation, object execution state, object communication mechanisms, etc. Following the CORBA architecture, most of the jobs to delivery object communication are transparent. In this sense, the four categories of OMA objects can be connected to a CORBA ORB to form a distributed computing environment without worrying about any of the communication issues among them. Above Figure demonstrates CORBA ORB architecture with its key components. Here, we try to understand CORBA structure through studying some of its key components. Some features that are important to CORBA are also discussed below.

3.2 DCOM

DCOM is more or less an architecture specification designed to be language-independent. DCOM is primarily implemented on Windows platforms, and specified by Microsoft [10]. Microsoft DCOM is often called ``COM on the wire". It supports remote objects by running on a protocol called *Object Remote Procedure Call* (ORPC).

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. 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 implemented on the platform. However, it is practically not available except Windows systems.

3.3 RMI

Sun Java RMI is a built-in native ORB in Java language. It supports making method invocations on remote objects. From practical programming point of view, developing distributed applications in RMI is simpler than developing with sockets since there is no need to design a protocol, which is an error-prone task. In RMI, the developer has the illusion of calling a local method from a local class file, when in fact the arguments are shipped to the remote target and interpreted, and the results are sent back to the callers. The underlying protocol for RMI is *Java Remote Method Protocol* (JRMP).

3.4 Feature Analysis and Semantic Comparison

As distributed object technologies the architectures of CORBA, DCOM and Java/RMI provide mechanisms for transparent invocation and accessing of remote distributed objects. Though their objective and approach is more or less same the mechanisms that they employ to achieve remoting and many other issues with respect to their central objective is a lot different. The following table provides detailed comparisons based on different aspects.

	DCOM	CORBA	Java/RMI
Base Interface (Base Type)	Every sever object implements IUnknown interface	Every sever object implements CORBA.Object	Every server object implements java.rmi.Remote
IDL	CORBA IDL defines the methods and attributes of component interface The IDL compiler creates proxy stubs for the client and server.	The Microsoft's MIDL DCOM interfaces. The MIDL compiler creates proxy stubs for the client and server.	RMI defines interfaces in Java language. RMIC compiler is used tocompile and create proxy stub and skeleton
Unique Identification	interface – an interface is uniquely identified by an id called IID named implementation of the server object is uniquely identified by id called CLSID	interface – an interface is uniquely identified by an 'interface name' named implementation of server object is uniquely identified by its mapping to a name in the Implementation	interface – an interface is uniquely identified by an interface named implementation of the server object is uniquely identified by its mapping to a URL in the Registry

		Repository	
Remote Object Reference (object handle at run-time)	Uniquely identifies a remote server object through its interface pointer, which serves as the object handle at run-time.	Uniquely identifies remote server objects through object references(objref), which serves as the object handle at run-time	Uniquely identifies remote server objects with the ObjID, which serves as the object handle at run-time.
Object Handle	interface pointer	Object Reference	Object Reference
Remote Object Reference Creation	Generated by Object Exporter	Generated by the Object Adapter	Generated by the call to the method Unicast Remote Object. Export Object (this)
object and skeleton instantiation	Tasks like Object and skeleton registration are performed by server program or handled dynamically by the COM run-time system.	Tasks like Object and skeleton registration are performed by	object registration is done through RMI Registry using Naming class. skeleton registration is done by its instantiation on calling UnicastRemoteObject.exportObject(this) method
Underlying Remoting Protocol	Object Remote Procedure Call(ORPC)	Internet Inter-ORB Protocol(IIOP)	Java Remote Method Protocol (JRMP)
Object Activation	Client calls CoCreateInstance()it needs a server object	Client binds to a naming or a trader service when it needs server object	Client calls lookup() on the remote server object's URL name when it needs server object
Mapping of Object name to Object Implementation	Handled by the windows Registry	handled by the Implementation Repository	Handled by the RMIRegistry
Type Information for methods locating an object implementation	Stored in Type Library Handled by Service Control Manager (SCM)	Stored in Interface Repository Handled using Object Request Broker (ORB)	Any type information is held by the Object itself handled Java Virtual Machine (JVM)
Activating object implementation	Handled by Service Control Manager (SCM)	Handled either by Basic Object Adapter (BOA) or Portable Object Adapter (POA)	Handled by Java Virtual Machine (JVM)
Parameter passing	All parameters passed between the client and server objects are passed either by value or by reference.	All interface types are passed by reference. All other objects are passed by value including highly complex data types	All objects implementing ‘remote interfaces’ extending java.rmi.Remote are passed by remote reference. All other objects are passed by value
Parameter Marshalling	Parameter marshalling is accomplished in the stub code that is generated by the IDL compiler. The client stub and server skeleton are responsible for marshalling of parameters.	DCOM provides automatic marshalling for primitive types and object references. For user defined structures and structured arrays Custom marshalling is preferred.	RMI provides automatic marshalling of predefined types and object references Serialization is used for marshalling objects.
Platform Independence	Runs on any platform having a COM Service implementation available on it.	Runs on any platform having CORBA ORB implementation available on it.	Runs on any platform having Java Virtual Machine implementation available on it
Language Independence	Yes	Yes	No. Only Java
Exception Handling	Through HRESULT return status, and Error Objects of type IErrorInfo and server object implementing ISupportErrorInfointerface.	Through Exception Objects	Through RemoteException
Support for data and code reuse	Supports code reuse by writing CORBA compatible new objects.	Supports code reuse just by modifying the registry entry, without needing to recompile code on the client or server.	Supports code reuse through Object Inheritense
Support for Multiple Inheritacne	Yes, at Interfaces as well as Object implementation.	Yes, at interface level	Yes, at interface level
Security	Provided by CORBA Security	Provided by NT security	RMI security is provided by java security API.

4. Conclusion

The Distributed Object Technologies facilitate methods of an object active on one machine to be accessed by remotely located programs in a smooth, secure and transparent manner. The two communication parties called client and server programmes may be running on heterogeneous platform both in terms of hardware as well as software. This paper presented a analytical survey of most popular Distributed Object Technologies, CORBA, DCOM and RMI. The survey conducted with respect to aspects like platform independence, language independence, marshalling, reusability, remoting protocol, security etc. and a comparison is presented which helps choosing technology most suitable for one’s application.

References

[1] Berners-Lee, Tim. "Hyper Text Transfer Protocol". World Wide Web Consortium. Retrieved 31 August 2010.
[2] Prince, Brian. "Should Organizations Retire FTP for Security?". *Security Week*. Security Week. Retrieved 14 September 2017.
[3] Dustdar, S.; Schreiner, W. (2005). "A Survey on web services composition" . *International Journal of Web and Grid Services*. IJWGS.2005.007545
[4] "Distributed Application Architecture " (PDF). Sun Microsystem. Archived from the original (PDF) on 6 April 2011. Retrieved 2009-06-16.
[5] Herbert P. Luckett, What's News: Electronic-mail delivery gets started, Popular ScienceArchived 2016-04-30
[6] Conrad, Jennifer (2003). "Institutional Trading and Alternative Trading Systems". *Journal of Financial Economics*
[7] "IRC Chatiquette–Chat Etiquette". Livinginternet.com. 28 November 1995. Retrieved 19 January 2012.
[8] Arpaci-Dusseau, Remzi H.; Arpaci-Dusseau, Andrea C. (2014), *Introduction to Distributed Systems* (PDF), Arpaci-Dusseau Books.
[9] "RMI Unleashes the Highest Performing Multi-core Processor and Product Family in the Industry, Driving System and Performance Scalability". *Press release*. RMI. May 19, 2009.
[10] J. D. Schoeffler, "A Model For Estimating Overhead in DCOM and CORBA Function Calls", NASA Report, 1998
[11] Elfwing, R., Paulsson, U., and Lundberg, L,*Performance of SOAP in Web Service Environment Compared to CORBA,* In Proceedings of the Ninth Asia-Pacific Software Engineering Conference, IEEE, 2002