ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website:www.ijircce.com

Vol. 5, Issue 2, February 2017

# Survey on Interprocess Communication and Management

Snehal Ghodake, Prof. A. R. Buchade

M.E. Student, Department of Computer Engineering, Pune Institude of Computer Technology, Pune, India

Assistant Professor, Department of Computer Engineering, Pune Institude of Computer Technology, Pune, India

**ABSTRACT:** Interprocess communication (IPC) refers to the coordination of activities among cooperating processes. IPC is the heart of all distributed systems, so we need to know the ways that processes can exchange the information. Communication middleware is computer software that enables two or more separate software components, processes, and/or applications to exchange information, either within one device, or between multiple devices. Often, this is referred to as IPC. Communications middleware is an IPC mechanism, which is responsible for controlling IPC activities. This paper will explore various mechanisms of interprocess communication.

**KEYWORDS**: Distributed Computing, Interprocess communication, CORBA, Socket, RPC, REST

## I. INTRODUCTION

Middleware is the software that enables communication and management of data in distributed applications. Businesses frequently need middleware like applications to link information from departmental databases, such as payroll, sales, and accounting, or databases housed in multiple geographic locations. There is a need to develop a middleware like framework[11] for an enterprise application for communication between client and server. It enables interoperability between processes that run on different operating systems(Windows,Linux), by supplying services so the processes can exchange data in a standards-based way. It provides unique communication interface for every control process, by serving as an independent programming interface for their applications.

## II. RELATED WORK

Following are the ways to achieve interprocess communication in distributed system:

1) CORBA
2) Socket Communication
3) RPC
4) REST based communication

1. CORBA:

The Common Object Request Broker Architecture (CORBA) is a standard developed by the Object Management Group (OMG) to provide interoperability among distributed objects[6]. CORBA is the world's leading middleware solution enabling the exchange of information, independent of hardware platforms, programming languages, and operating systems. CORBA uses an interface definition language (IDL) to specify the interfaces that objects present to the outer world. CORBA then specifies a mapping from IDL to a specific implementation language like C++ or Java.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website:www.ijircce.com

Vol. 5, Issue 2, February 2017

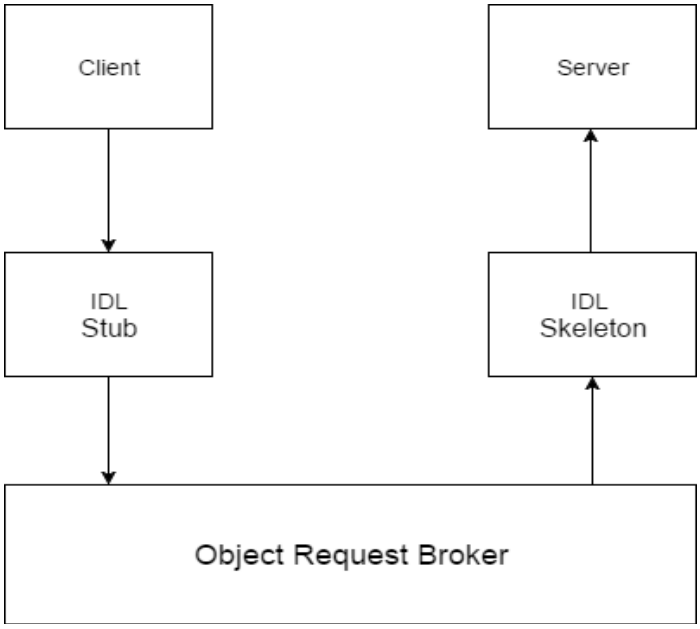


Fig. 1 Architecture of CORBA

Figure 1 shows the architecture of CORBA:

1) Client : This is the program entity that invokes an operation on an object implementation. Accessing the services of a remote object should be transparent to the caller. Ideally, it should be as simple as calling a method on an object.
2) Server : This is the program entity that provides service to clients.
3) IDL stub and skeleton : CORBA IDL stubs and skeletons serve as the glue between the client and server applications, respectively, and the ORB. The transformation between CORBA IDL definitions and the target programming language is automated by a CORBA IDL compiler. The use of a compiler reduces the potential for inconsistencies between client stubs and server skeletons and increases opportunities for automated compiler optimizations.
4) Object Request Broker : The ORB provides a mechanism for transparently communicating client requests to target object implementations. The ORB simplifies distributed programming by decoupling the client from the details of the method invocations. This makes client requests appear to be local procedure calls. When a client invokes an operation, the ORB is responsible for finding the object implementation, transparently activating it if necessary, delivering the request to the object, and returning any response to the caller.

2. Socket communication:

Another solution is to use socket-based network communication directly between the applications. This is a relatively low-level approach. Sockets allow communication between two different processes on the same or different machines[1]. With XML as the message protocol between them, we can maintain a degree of platform and language independence. It is a way to talk to other computers using standard file descriptors. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website:www.ijircce.com

Vol. 5, Issue 2, February 2017

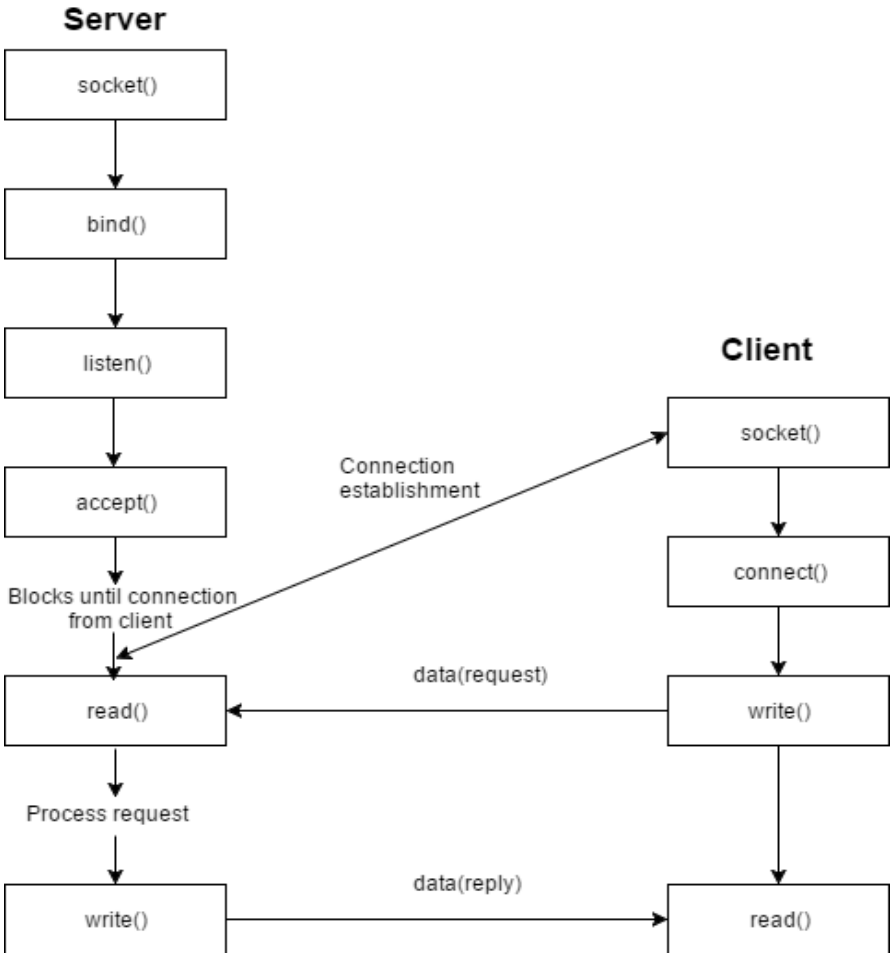


Fig. 2 Flow of socket communication

Figure 2 shows the flow of communication between server and client:

1) Server and client create a stream socket with the socket() call.
2) (Optional for client) Sever bind socket to a local address with the bind() call.
3) Server uses the listen() call to alert the TCP/IP machine of the willingness to accept connections.
4) Client connects socket to a foreign host with the connect() call.
5) Server accepts the connection and receives a second socket, with the accept() call.
6) Server reads and writes data on socket , client reads and writes data on socket by using read() and write() calls, until all data has been exchanged.

3. RPC:

Remote Procedure Call (RPC) is a powerful technique for constructing distributed, client-server based applications. It is based on extending the notion of conventional, or local procedure calling, so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them. By using RPC, programmers of distributed applications avoid the details of the interface with the network. The transport independence of RPC isolates the application from the physical and logical elements of the data communications mechanism and allows the application to use a variety of transports. RPC makes the client/server model of computing more powerful and easier to program.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website:www.ijircce.com

Vol. 5, Issue 2, February 2017

To develop an RPC application the following steps are needed:

1) Specify the protocol for client server communication
2) Develop the client program
3) Develop the server program

The programs will be compiled separately. The communication protocol is achieved by generated stubs and these stubs and rpc (and other libraries) will need to be linked in.

XML-RPC is a remote procedure call protocol which uses XML to encode its calls and HTTP as a transport mechanism[15]. XML-RPC also refers generically to the use of XML for remote procedure call, independently of the specific protocol. XML-RPC works by sending an HTTP request to a server implementing the protocol. The client in that case is typically software wanting to call a single method of a remote system. Multiple input parameters can be passed to the remote method, one return value is returned. The parameter types allow nesting of parameters into maps and lists, thus larger structures can be transported. Therefore, XML-RPC can be used to transport objects or structures both as input and as output parameters.

4. REST based communication:

Representational state transfer (REST) or RESTful Web services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate web resources using a uniform and predefined set of stateless operations. REST defines a set of architectural principles by which we can design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. There are various ways of developing RESTful services in Java like using JAX-RS and Jersey, or using annotations in Spring Boot MVC architecture. But, developing RESTful services in C++ application is quite challenging. By using the C++ REST SDK (codename "Casablanca")[4, 5], we can more easily write modern, asynchronous C++ code that connects with REST-based services. In this approach we are using JSON for communication. JSON (JavaScript Object Notation) is a lightweight data-interchange format[7]. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is a text format that is completely language independent.

JSON is built on two structures:

1) A collection of name/value pairs.
2) An ordered list of values.

RESTful applications use HTTP requests to create/update data, to read data and to delete data. Thus, REST uses HTTP for all four CRUD (Create/ Read/Update/ Delete) operations. In REST each data element is a resource, addressed by a URI (Uniform Resource Identifier).

## III. ADVANTAGES AND DISADVANTAGES

In above section, we have seen various ways of interprocess communication between processes. Although, all are suitable they have some pros and cons.

1. Socket

1) Advantages

Sockets are flexible and sufficient. Sockets provide the communication mechanism between two computers using TCP. Efficient socket based programming can be easily implemented for general communications[12]. Sockets cause low network traffic.

2) Disadvantages

It can establish communication only with the machine requested and not with any other machine on the network. Sockets allow only raw data to be sent. This means that both client and server need to have mechanisms to interpret the data.

2. CORBA

1) Advantages

CORBA is extremely feature-rich, supporting many programming languages, operating systems, and a diverse range of capabilities such as transactions, security, Naming and Trading services[6].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website:www.ijircce.com

Vol. 5, Issue 2, February 2017

2) Disadvantages

There's no real CORBA standard to bind an ORB and its clients to a port or a port range, there are (only) vendor specific options.The weakness of CORBA lies in its specifications and protocols are extremely large and complex. To adopt CORBA technology, the programmer must face the complexity of mapping from the IDL to the implementation-language data types[2].

3. RPC

1) Advantages

RPC is Server independent. Process-oriented and thread oriented models supported by RPC. The development of distributed systems is simple because it uses straightforward semantics and easier. The code re-writing / re-developing effort is minimized. Enables the usage of the applications used in the distributed environment, not only in the localenvironment.

2) Disadvantages

RPC is not a standard – it is an idea that can be implemented in many ways. RPC does not solve the most of the distribution creation problems. RPC is only interaction based. This does not offer any flexibility in terms of hardware architecture.

4. REST

REST is primarily used to build Web services that are lightweight, maintainable, and scalable. A service based on REST is called a RESTfulservice[10]. Every system uses resources. These resources can be pictures, video files, Web pages, business information, or anything that can be represented in a computer-based system. The purpose of a service is to provide a window to its clients so that they can access these resources. Service architects and developers want this service to be easy to implement, maintainable, extensible, and scalable. A RESTful design promises that and more.

## IV. CONCLUSION

Client server communication in distributed environment is essential requirement now a days. Clients and servers may be residing on same or different machines over the network. As we have seen there are various ways to achieve interprocess communication in distributed system with their pros and cons. REST API based communication is efficient among all. It helps to achieve this with minimum overhead. By simply giving the URLs and hitting REST end points we can access the services.

## REFERENCES

1. Peter A. Cooper, "Building client-server systems: exercises in network communication, ACM,Journal of Computing Sciences in Colleges: Volume 15 Issue 3, 2000.
2. Wang Shaofeng, Sun Jiaguang, "A framework design of workflow management system with Java RMI", ACM SIGPLAN Notices: Volume 36 Issue 9, 2001.
3. M A Hossain and M O Tokhi, "INTER-PROCESSOR AND INTER-PROCESS COMMUNICATION IN REALTIME MULTI-PROCESS COMPUTING", Elsevier, IFAC 15th Triennial World Congress, Barcelona, Spain, 2002.
4. https://msdn.microsoft.com/en-us/library/jj969455.aspx
5. https://casablanca.codeplex.com/
6. Wang Zhaoshun; Yingjian An; Li Tao, " Research of software component development based on CORBA",2010 2nd IEEE International Conference on Information Management and Engineering,2010.
7. Philipp Wehner, Christina Piberger, Diana Göhringer, " Using JSON to manage communication between services in the Internet of Things", IEEE, 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC),2014.
8. Yao Zhao, Li Dong, Rongheng Lin, Danfeng Yan, Jun Li, "Towards Effectively Identifying RESTful Web Services", IEEE International Conference on Web Services,2014.
9. Florian Haupt, Frank Leymann,CesarePautasso, "A conversation based approach for modeling REST APIs", 12th Working IEEE/IFIP Conference on Software Architecture, 2015.
10. UrjitaThakar,AmitTiwari,SudarshanVarma, "On Composition of SOAP Based and RESTful Services", IEEE 6th International Conference on Advanced Computing, 2016.
11. Ting-HuanKuo, Chi-Hua Chen, "Applications of the Web Service Middleware Framework Based on the BPEL", IEEE 5th Global Conference on Consumer Electronics, 2016.
12. UNIX Network Programming Book by W. Richard Stevens.
13. Pascal Giessler, DmitrijSarancin, "Best Practices for the Design of RESTful Web Services", The Tenth International Conference on Software Engineering Advances, 2015.
14. GuoliJi, Xiaorong Hu; Xiaozhong Li, Qianbin Huang, Meishuang Tang, "A web implementation of MPC system with model diagnosis and adjustment based on XML-RPC", Computer and Communications (ICCC), 2015 IEEE International Conference, 2015.