Alexandros Karakasidis

Assignment 4

This article presents and compares two of the most widely used frameworks for developing distributed applications: RPC and RMI.

RPC based mechanisms assumed a heterogeneous environment consisting of machines with different operating systems. Proxies are used in order to deal with heterogeneity: their client side is called stub, and the server side is called skeleton. Marshalling and unmarshaling was necessary in order to adapt the different message representations to a neutral data representation. These proxies were implemented using language and machine-neutral interface-definition language descriptions called IDLS. The limitations posed by such representations mainly affected the variation of data kinds that can travel between different machines. However, the greatest limitation of this approach, was its static nature, since the two processes have to explicitly know what kind of data is exchanged through the network.

On the other hand, Java RMI is built on the assumption that all objects are written in Java and each machine can offer the same execution environment. This assumption allows every Java object to pass as a function parameter. Furthermore, since this approach is based on Java, makes possible the assumption of dynamic code loading on the JVM. Remote objects pass by reference and local objects pass by value. This allowed by sophisticated mechanisms, which can handle inheritance and polymorphism issues. The way stubs are created, allows them to provide interfaces to the whole set of methods the remote object supports. In conjunction with the dynamic code loading ability, it is made possible to reduce remote calls with caching techniques using proxies.

Concluding, the fact that the interface implementation comes from the remote object itself, and the ability of dynamic class loading, make the most fundamental differences between RPC and RMI.