Chapter 1

Conclusion

1.1 Conclusion

Java RMI (Remote Method Invocation) can be used to invoke methods of objects in another Java Virtual Machine (JVM). The JVM can be on the local host or another host. RMI allows the client to invoke remote methods in the server as if the remote object were contained on the client host, from the programmer's perspective. This separation between interfaces and the objects implementing these interfaces allows developers to place an interface on one node, while the object itself resides on another node. When a client binds to a distributed object, an implementation of the object's interface - called a proxy, is then loaded into the clients address space making it available in method-invocations.
When a client invokes a method on a remote object, instead of letting the Java runtime take care of the call as it would for regular Java objects, the RMI runtime system takes over and routes the call over the network to the remote object. None of the underlying communication is visible to the programmer.

Java RMI can be used in distributed systems where multiple PC's are running multiple processes and theses processes have to cooperate in solving a given task. Handling all these processes in cooperating, its important that there is a leader, which all process can interact with. To find a leader between all the processes, there are different leader election algorithms which could be used, e.g. Bully algorithm or The Chang and Roberts algorithm. The leader election algorithms are used when the systems starts and when a leader process have failed or retires on purpose. This mean that the system can go on, even if a given leader process is failing.

1.2 Discussion

The Java RMI is simple and easy to use as the application makes services and resources on servers available by offering an interface. This ease the sharing of these services in distributed systems as it offers access to the remote object as if they were local. The underlying network connection is handled by the application, which makes it easy to implement. However it is important to consider that the application only support Java-to-Java collaboration. But if wanted a Java Native Interface can be used to map between software written in another programming language and the RMI on the client space. This enables the use of Java RMI even if the Java programming language is not used. If this is the circumstance it could be feasible to use CORBA instead, which support multiple programming languages. This interface is offering a variety of complicated methods which is not always an advantageous, as a client rarely use all methods. Thus a simple method-invocation can be unnecessarily large to implement
It is also to be considered that large network-collaborations can experience delays as the remote method invocations are more time consuming than local methods invocations. Furthermore the need of at least two JVMs is also considered to be time consuming. Therefore it is advisable to implement time-out handlers on the clients space. If the server-side fails, then the client would handle the exception when the time has run out.
Also when using the Bully election a new election can be quite consuming. But the use of a leader is on the same time essential, as the the nodes in the network are collaborating in achieving a common goal. This require a common understanding on how and when the processes can perform a given task, which is supplied by the leader. But even with a leader to coordinate the remote calls may introduce concurrency issues, which should be handled by the client.

1.3 Perspectives

The Java RMI application can be used when the Client-Server architecture is wanted, that be the server handling big calculation-tasks or database-communication and the client using these services/resources. Concrete it could be feasible when personal healthcare devices need to deliver healthcare-data to a manager-node which handle the database communication. The data-transfer is handled by Java RMI, which make it easier to implement for the developer, since the focus only have to be on the client-server relationship and e.g. not network transfer.