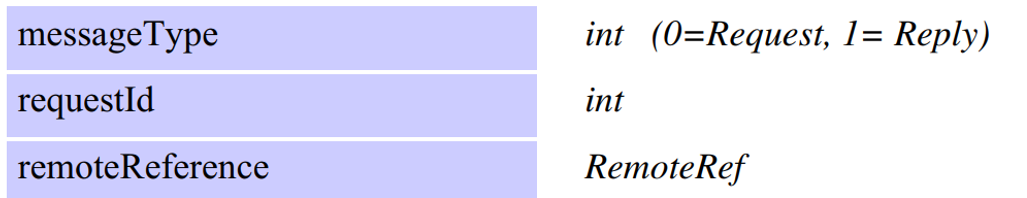


Communication module: The two cooperating communication modules carry out the request-reply protocol

The communication module uses the first three items



1 The communication modules are together responsible for providing a specified invocation semantics, for example at-most-once or at-most-once. Means that the caller either receives a result or an exception. (The remote reference of the object to be invoked.)

The communication module in the server selects the dispatcher for the class of the object to be invoked, passing on its local reference,

2 Remote reference module is responsible for translating between local and remote object references and for creating remote object references(has a remote object table)

The table includes:

An entry for all the remote objects held by the process.

An entry for each local proxy.

Servants • A servant is an instance of a class that provides the body of a remote object

Proxy: The role of a proxy is to make remote method invocation transparent to clients by behaving like a local object to the invoker(marshalling and unmarshalling)

There is one proxy for each remote object for which a process holds a remote object reference

The class of a proxy implements the methods in the remote interface of the remote object it represents.

Each method of the proxy marshals a reference to the target object, its own operationId and its arguments into a request message and sends it to the target

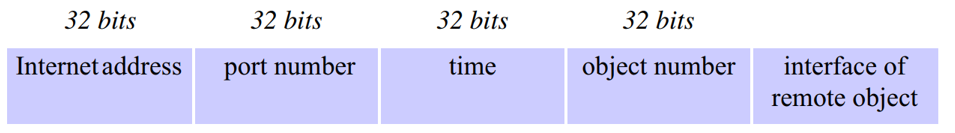
Dispatcher: A server has one dispatcher and one skeleton for each class representing a remote object

the dispatcher receives request messages from the communication module. It uses the operationId to select the appropriate method in the skeleton, passing on the request message.

Skeleton: The class of a remote object has a skeleton, which implements the methods in the remote interface.

A skeleton method unmarshals the arguments in the request message and invokes the corresponding method in the servant.

Remote object references are marshalled in this way



Parameter and result passing

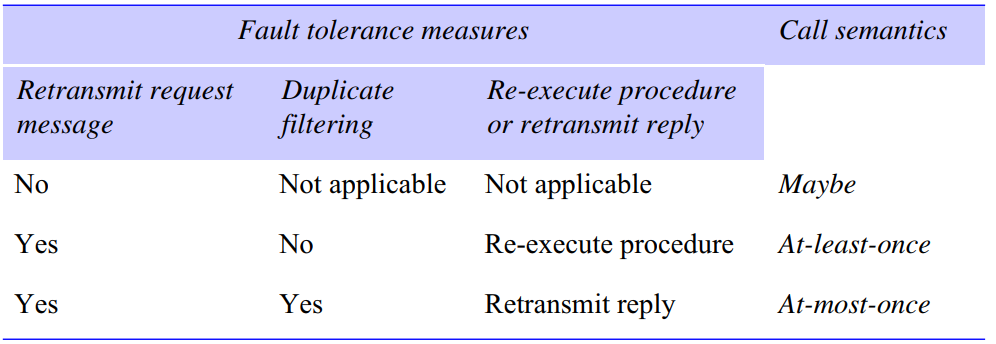
•In Java RMI, the parameters of a method are assumed to be input parameters and the result of a method is a single output parameter(all argument that will be passed throw RMI should be serializable, all primitive types are serializable )

Passing remote objects: When the type of a parameter or result value is defined as a remote interface, the corresponding argument or result is always passed as a remote object reference

Passing non-remote objects: All serializable non-remote objects are copied and passed by value

When any object is serialized, its class information is annotated with the location of the class (as a URL), enabling the class to be downloaded by the receiver.

Virtual machine in the receiving process will download.



Retry request message: Controls whether to retransmit the request message until either a reply is received or the server is assumed to have failed.

Duplicate filtering: Controls when retransmissions are used and whether to filter out duplicate requests at the server.

Retransmission of results: Controls whether to keep a history of result messages to enable lost results to be retransmitted without re-executing the operations at the server.

Transparency: aimed to make remote procedure calls as much like local procedure calls as possible