3 Remote Invocation / RMI

Saturday, January 12, 2019 2:04 PM

Give an overview of Java-RMI. Describe parameter passing in Java-RMI. Discuss some of the issues with remote invocation including, but not limited to, parameter passing methods.

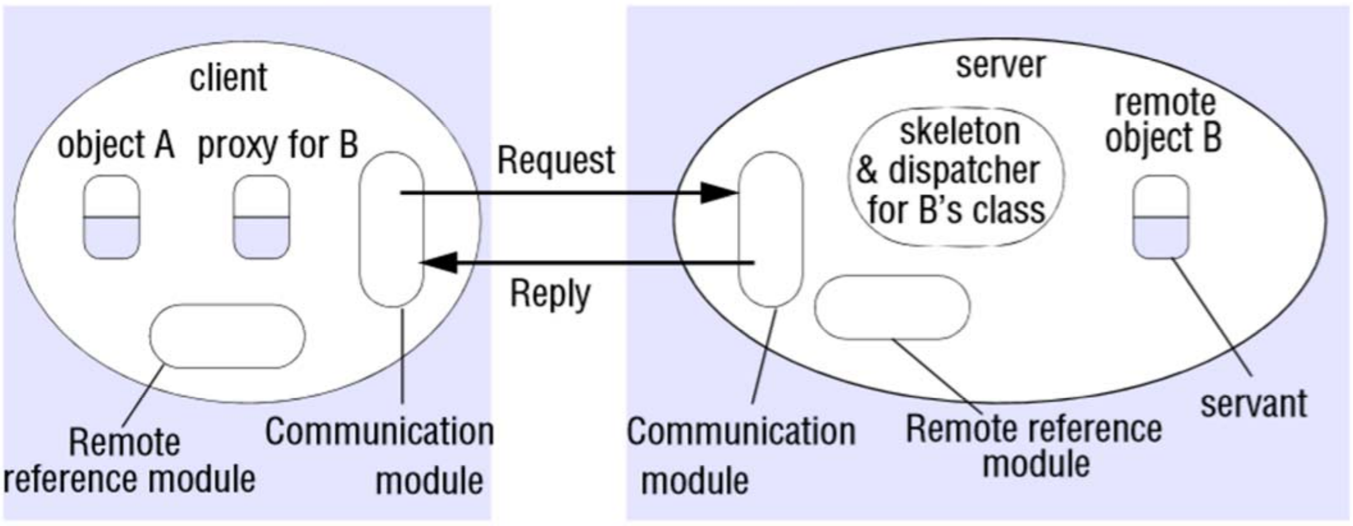
- Theory bhind rmi: marshalling / unmarshalling
- Java vs manual rmi

- RMI: has to affect an object
- RPC: doesn't

Generic RMI

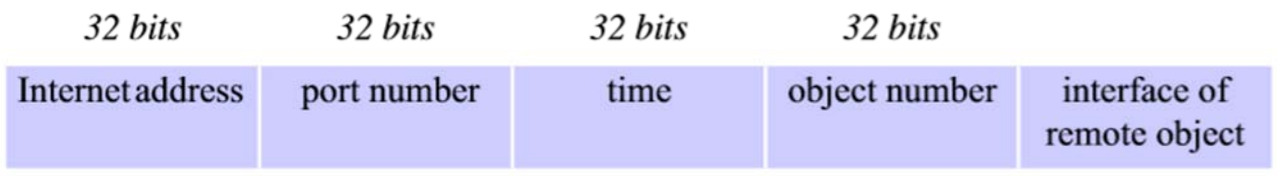
- Encapsulation: Object data is accessible indirectly through methods only
- Accessing objects via obj references
- The receiver (server) executes the actions and then returns control to invoker

- Define remote interface with CORBA



- Remote ref module: translates between local and remote obj references. When a new object reference arrives, it creates a new local proxy for that object.
    - Called when marshalling / unmarshalling requests
- Servant: the true instance that provides the body of the remote object
- The RMI Implementation
    - Proxy: behaves like a local object, but forwards methods; hides all the marshalling / communications jazz
        - Awaits reply, unmarshals and returns results to the invoker
    - Dispatcher: receives request, calls the implementation in the skeleton
    - Skeleton: implements the method of the remote interface.
        - Awaits result, then marshals to reply
- Binder: service; has mappings from textual names to object refeerences. Server can register remote objects by name and clients can look them up

Marshalled reference



32 bits	*32 bits*	*32 bits*	*32 bits*	
Internet address	port number	time	object number	interface of remote object

Java RMI

- Remote interfaces extend java.rmi.Remote
- Classes can be downloaded from one virtual machine to another if the object reference does not have the class for the proxy
- RMIregistry: the binder; should run on server

- Callbacks: way for server to notify clients that a change was made to the object (or that any event happened) instead of clients polling all the time