CLOUD COMPUTING APPLICATIONS

RPC IMPLEMENTATION

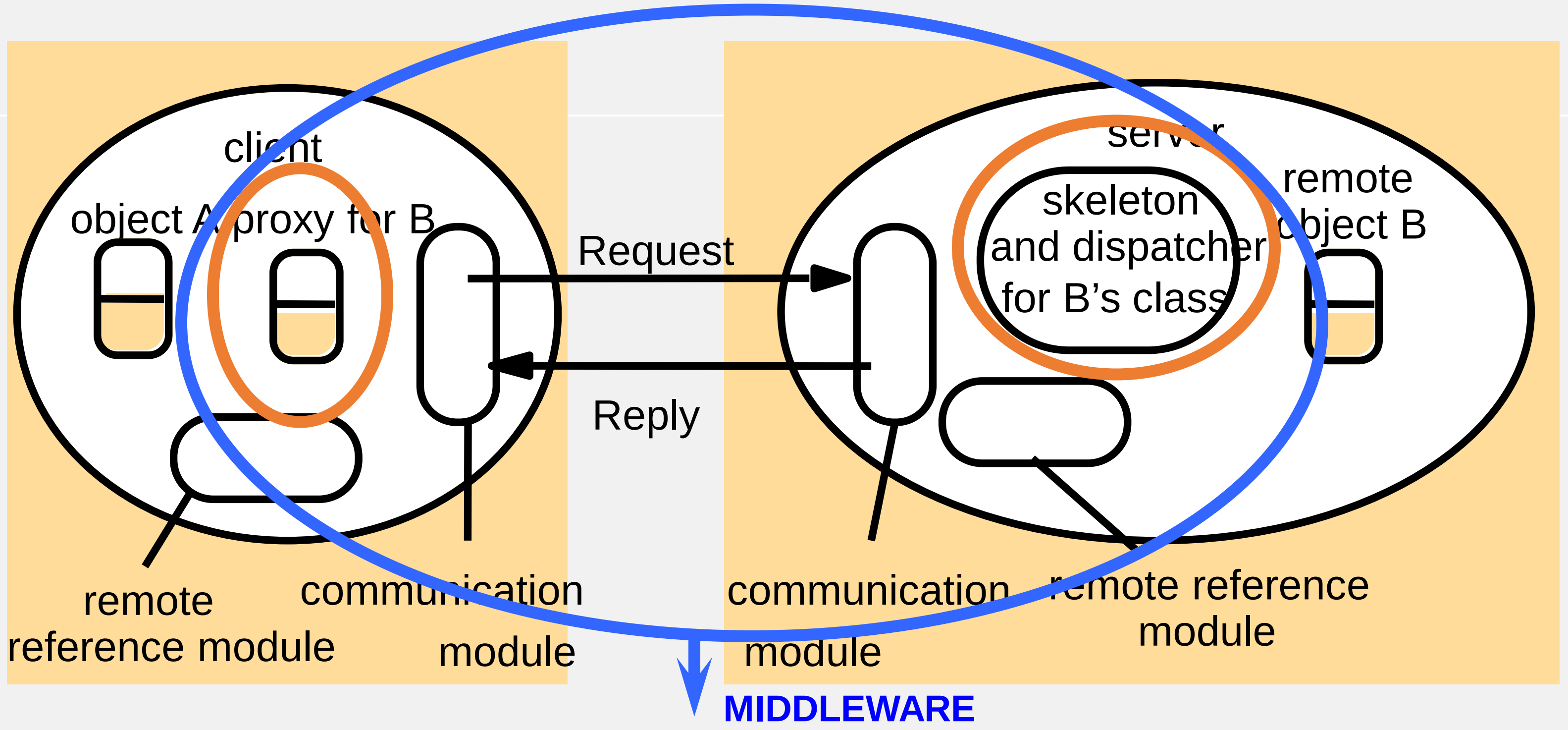
Prof. Roy Campbell

Contents

- Remote Procedure Calls
  - RMI, SOAP

How Do We Implement the Abstractions?

What should the middleware do to make the call appear similar to a local call?



- **Proxy on the "client" (process P1) side**
- **Skeleton and dispatcher on the "server" (process P2) side**

Proxy

- Is responsible for making RMI transparent to clients by behaving like a local object to the invoker
  - The proxy *implements* (Java term, not literally) the methods in the interface of the remote object that it represents. But…
- Instead of executing an invocation, the proxy forwards it to a remote object
  - On invocation, a method of the proxy *marshals* the following into a request message: (i) a reference to the target object, (ii) its own method id and (iii) the argument values. Request message is sent to the target, then proxy awaits the reply message, *unmarshals* it, and returns the results to the invoker
  - Invoked object unmarshals arguments from request message, and when done, marshals return values into reply message

Marshalling / Unmarshalling

- External data representation: an agreed, platform-independent, standard for the representation of data structures and primitive values
  - CORBA Common Data Representation (CDR)
  - Allows an ARM client (possibly big endian) to interact with a x86 Unix server (little endian).

- Marshalling: the act of taking a collection of data items (platform dependent) and assembling them into the external data representation (platform independent)

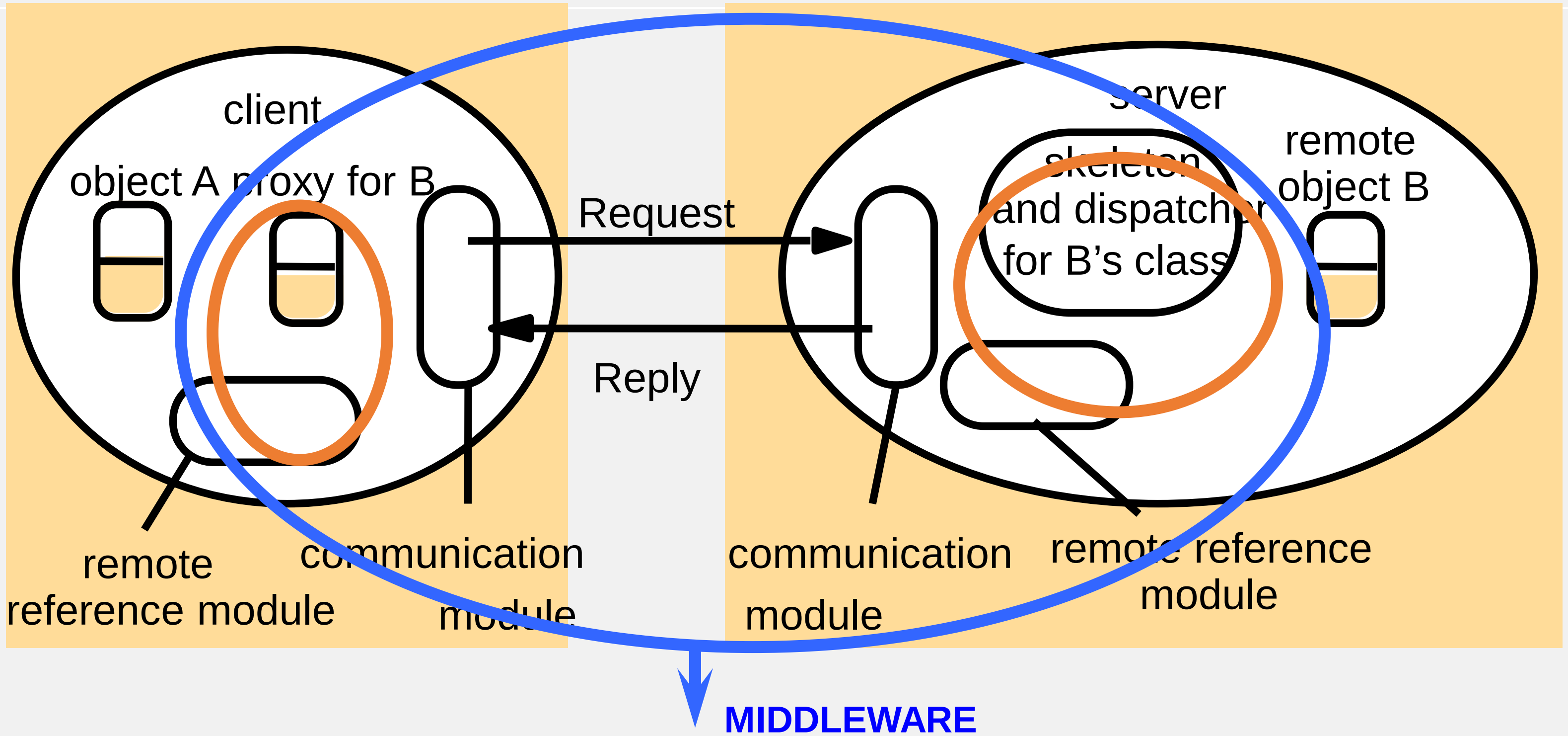
- Unmarshalling: the process of disassembling data that is in external data representation form into a locally interpretable form

Remote Reference Module

- Is responsible for translating between local and remote object references and for creating remote object references
- Has a *remote object table*
  - An entry for each remote object held by any process (e.g., B at P2)
  - An entry for each local proxy (e.g., proxy-B at P1)
- When the remote reference module sees a new remote object, it creates a remote object reference and adds it to the table
- When a remote object reference arrives in a request or reply message, the remote reference module is asked for the corresponding local object reference, which may refer to either a proxy or to a remote object
- In case the remote object reference is not in the table, the RMI software creates a new proxy and asks the remote reference module to add it to the table

How Do We Implement the Abstractions?

What should the middleware do to make the call appear similar to a local call?

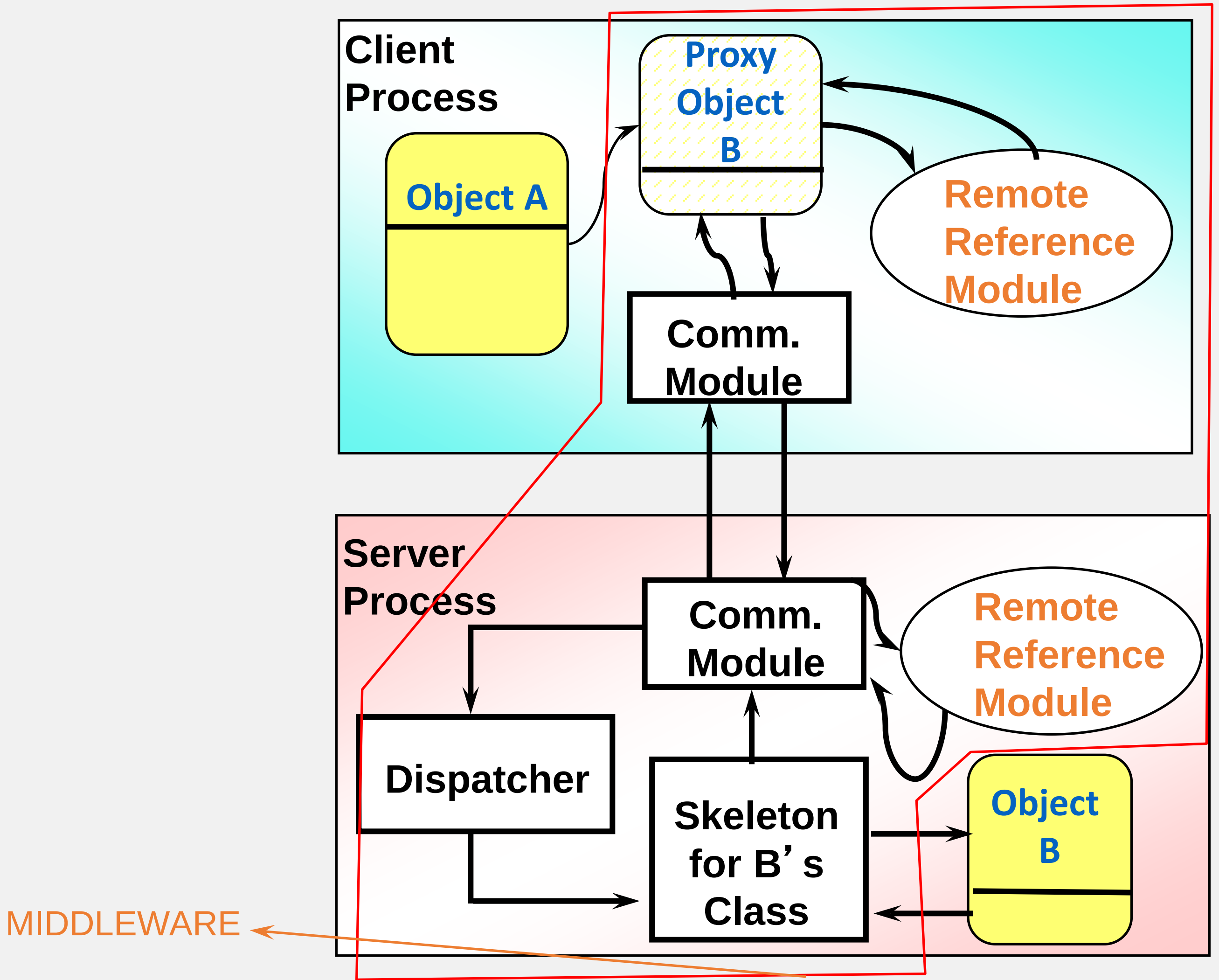


- **Proxy on the "client" (process P1) side**
- **Skeleton and dispatcher on the "server" (process P2) side**

What About Server Side? Dispatcher and Skeleton

- Each process has one dispatcher and a skeleton for each local object (actually, for the class)
- The dispatcher receives all request messages from the communication module
  - For the request message, it uses the method id to select the appropriate method in the appropriate skeleton, passing on the request message
- Skeleton "implements" the methods in the remote interface
  - A skeleton method unmarshals the arguments in the request message and invokes the corresponding method in the remote object (the actual object)
  - It waits for the invocation to complete and marshals the result, together with any exceptions, into a reply message

Summary of Remote Method Invocation (RMI)



Proxy object is a hollow container of method names

Remote Reference Module translates between local and remote object references

Dispatcher sends the request to Skeleton Object

Skeleton unmarshals parameters, sends it to the object, and marshals the results for return