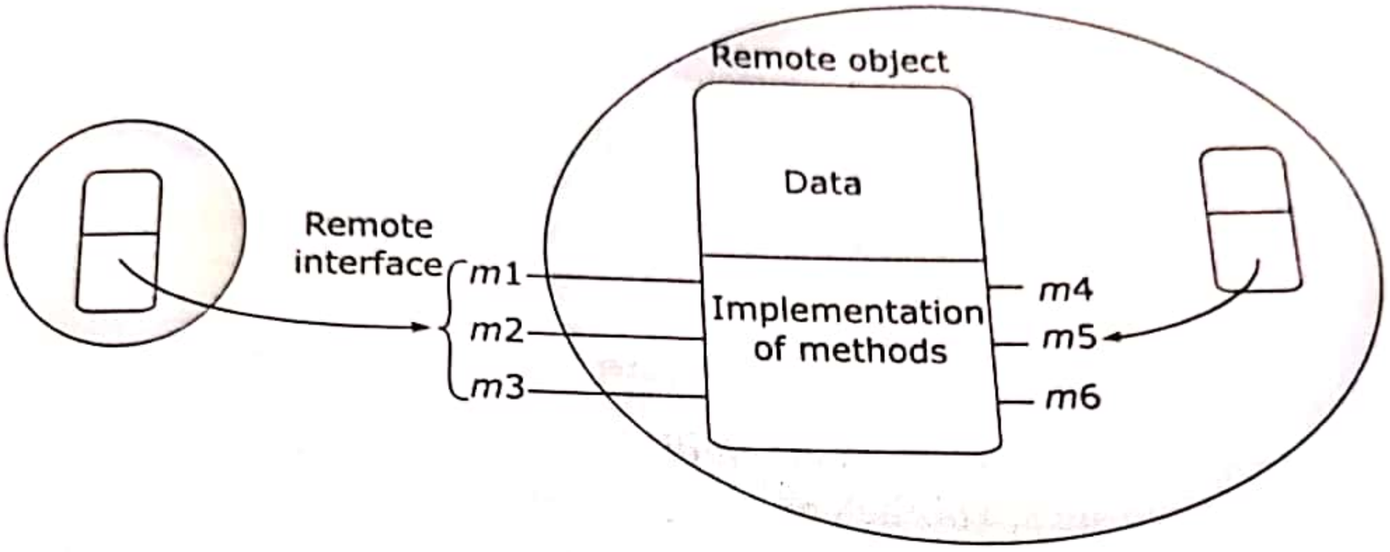


Figure 4-22 Remote object and remote interface

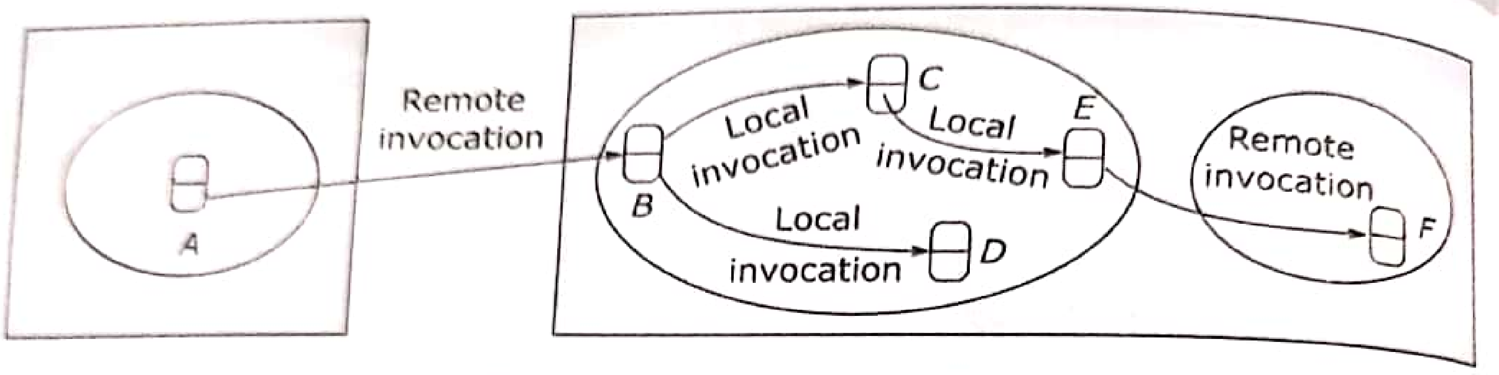


Figure 4-24 RMI and LMI

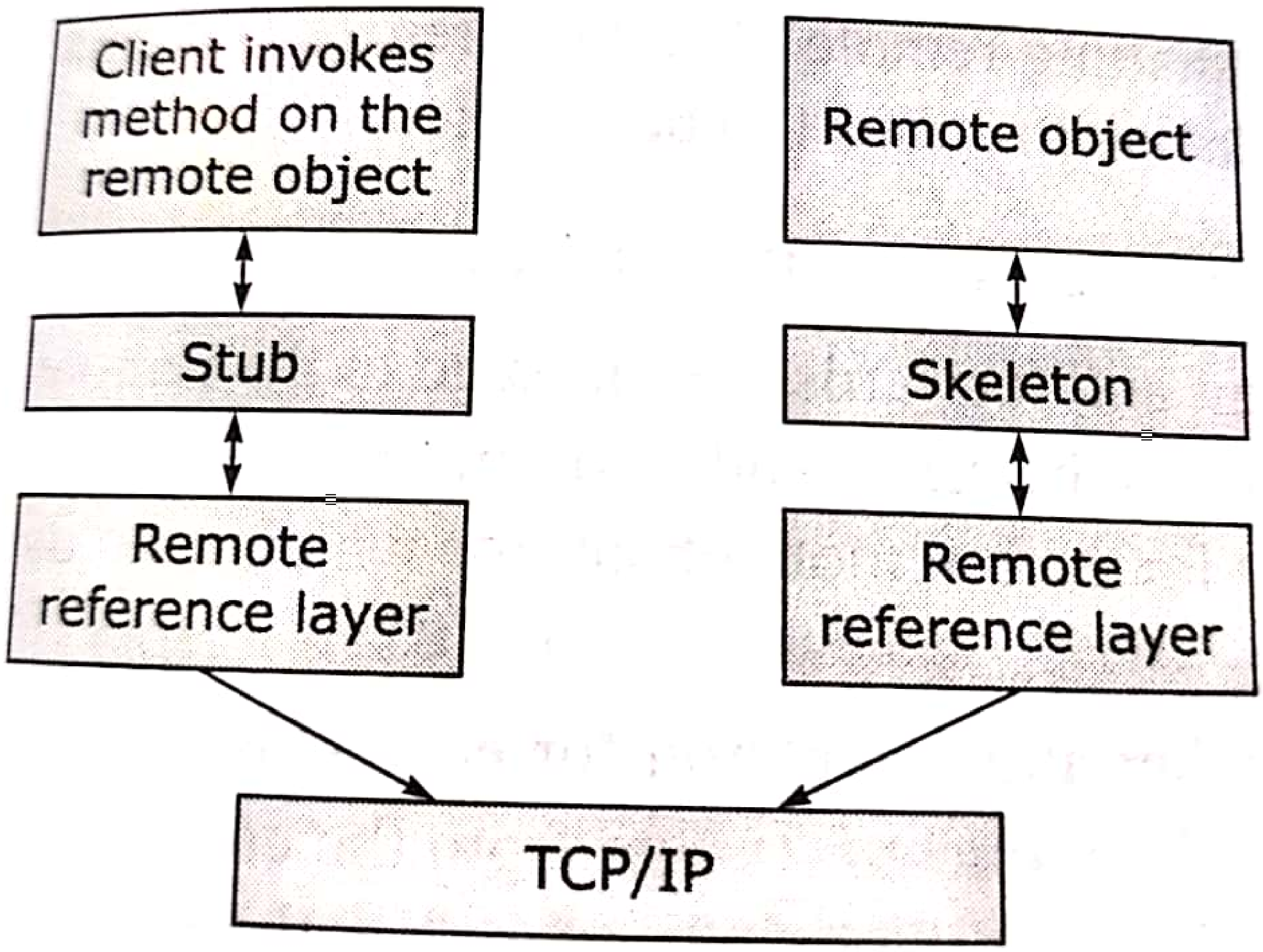


Figure 4-25 RMI flow diagram

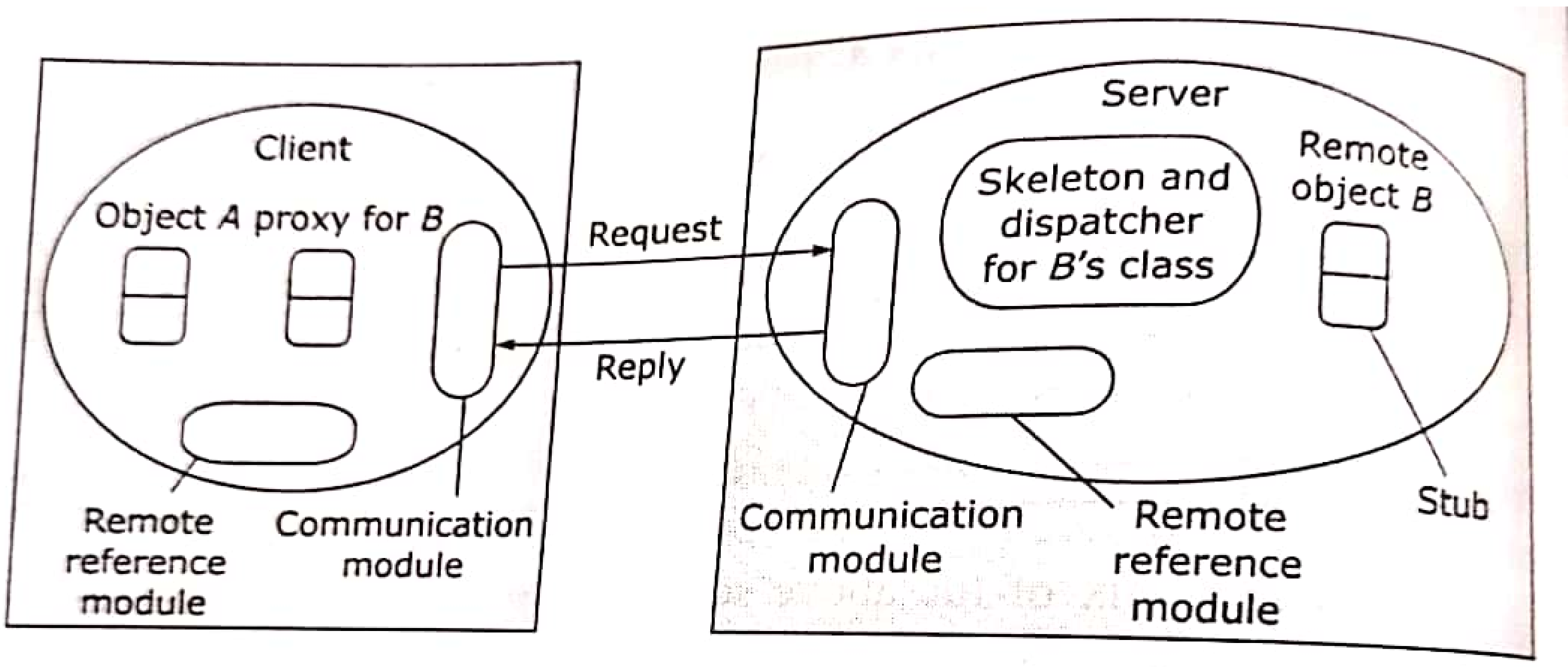


Figure 4-26 RMI components

4.7.2 RMI Execution

We now discuss the various components and the process of RMI execution, as shown in Figure 4-27. These are the communication module, remote reference module, RMI software, the server and the client programs, and the binder.

Communication module The client and the server processes form a part of the communication module, which use RR protocol. The format of the request-and-reply message is very similar to the RPC message.

Remote reference module It is responsible for translating between local and remote object references and creating remote object references. It uses a remote object table, which maps local to remote object references. This module is called by the components of RMI software for marshalling and unmarshalling remote object references. When a request message arrives, the table is used to locate the object to be invoked.

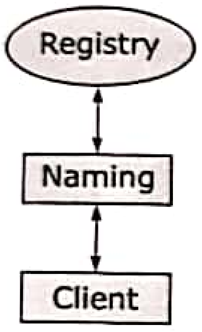
RMI software This is the middleware layer that consists of the following:

Proxy It makes RMI transparent to the client and forwards the message to the remot object. It marshals the arguments and unmarshals the result and also sends and receive messages from the client. There is one proxy for each remote object for which it hold a remote reference.

Dispatcher A server consists of one dispatcher and a skeleton for each class whic represents a remote object. This unit receives the request message from the commun cation module, selects the appropriate method in the skeleton, and passes the reque message.

Skeleton It implements the method in the remote interface. It unmarshals the arguments in the request message and invokes the corresponding method in the remote object. After the invocation is complete, it unmarshals the result along with excerptions in the reply message to send the proxy's method. The interface compiles automatically generating classes for proxy, dispatcher, and skeleton.

Server and client programs The server program contains classes for dispatcher, skeleton, and the remote objects it supports. A section creates and initializes at least one object hosted by the server. The other remote objects are created later when requests come from the client. The client program contains classes of processes for the entire remote object, which it will invoke. The client looks up the remote object reference using a binder.



e 4-28 Locating remote objects

The binder The concept of binders can be explained with an example. An object *A* requests remote object reference for object *B*. The binder in a distributed system is a service, which maintains a table of textual names to be remote object-referenced. Servers use this service to look up remote object references. Figure 4-28 shows how objects can be accessed remotely.