RPC Model

The remote procedure call (RPC) model is similar to a local procedure call model. In the local model, the caller places arguments to a procedure in a specified location such as a result register. Then, the caller transfers control to the procedure. The caller eventually regains control, extracts the results of the procedure, and continues execution.

RPC works in a similar manner, in that one thread of control winds logically through two processes: the caller process and the server process. First, the caller process sends a call message that includes the procedure parameters to the server process. Then, the caller process waits for a reply message (blocks). Next, a process on the server side, which is dormant until the arrival of the call message, extracts the procedure parameters, computes the results, and sends a reply message. The server waits for the next call message. Finally, a process on the caller receives the reply message, extracts the results of the procedure, and the caller resumes execution.

The **Remote Procedure Call Flow** figure illustrates the RPC paradigm.

In the RPC model, only one of the two processes is active at any given time. Furthermore, this model is only an example. The RPC protocol makes no restrictions on the concurrency model implemented, and others are possible. For example, an implementation can choose asynchronous Remote Procedure Calls so that the client can continue working while waiting for a reply from the server. Additionally, the server can create a task to process incoming requests and thereby remain free to receive other requests.

Transports and Semantics

The RPC protocol is independent of transport protocols. How a message is passed from one process to another makes no difference in RPC operations. The protocol deals only with the specification and interpretation of messages.

RPC does not try to implement any kind of reliability. The application must be aware of the type of transport protocol underneath RPC. If the application is running on top of a reliable transport, such as Transmission Control Protocol/Internet Protocol (TCP/IP), then most of the work is already done. If the application is running on top of a less-reliable transport, such as User Datagram Protocol (UDP), then the application must implement a retransmission and time-out policy, because RPC does not provide these services.

Due to transport independence, the RPC protocol does not attach specific semantics to the remote procedures or their execution. The semantics can be inferred from (and should be explicitly specified by) the underlying transport protocol. For example, consider RPC running on top of a transport such as UDP. If an application retransmits RPC messages after short time outs and receives no reply, the application infers that the procedure was executed zero or more times. If the application receives a reply, the application infers that the procedure was executed at least once.

A transaction ID is packaged with every RPC request. To ensure some degree of execute-at-most-once semantics, RPC allows a server to use the transaction ID to recall a previously granted request. The server can then refuse to grant that request again. The server is allowed to examine the transaction ID only as a test for equality. The RPC client mainly uses the transaction ID to match replies with requests. However, a client application can reuse a transaction ID when transmitting a request.

When using a reliable transport such as TCP/IP, the application can infer from a reply message that the procedure was executed exactly once. If the application receives no reply message, the application cannot assume that the remote procedure was not executed. Even if a connection-oriented protocol like TCP/IP is used, an application still needs time outs and reconnection to handle server crashes.

Transports besides datagram or connection-oriented protocols can also be used. For example, a request-reply protocol, such as Versatile Message Transaction Protocol (VMTP), is perhaps the most natural transport for RPC.

RPC in the Binding Process

The act of binding a client to a service is *not* part of the Remote Procedure Call specification. This important and necessary function is left to higher-level software. However, the higher level software may use RPC in the binding process. The RPC port mapper program is an example of software that uses RPC.

The RPC protocol's relationship to the binding software is similar to the relationship of the network jump-subroutine instruction (JSR) to the loader (binder). The loader uses JSR to accomplish its task. Similarly, the network uses RPC to accomplish the bind.
