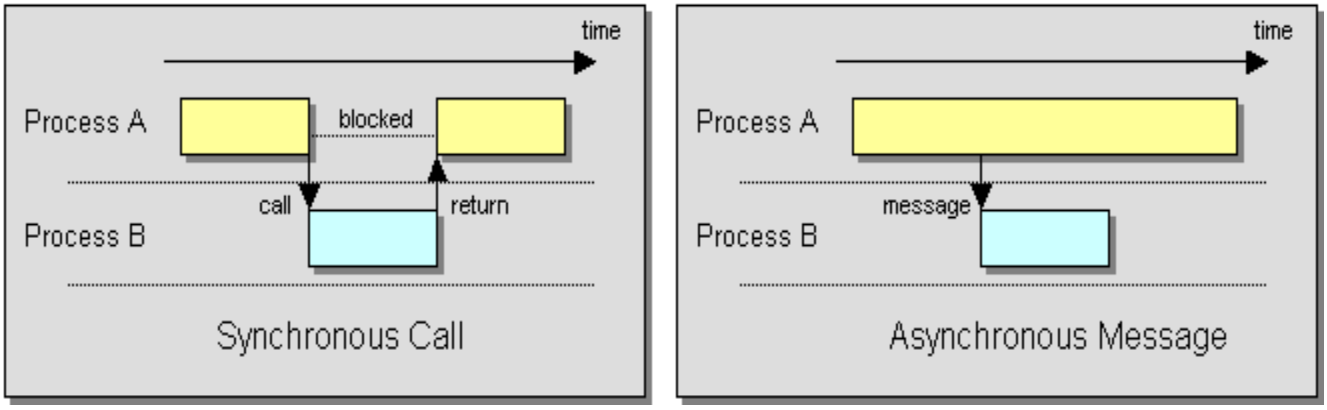
THINKING ASYNCHRONOUSLY

Messaging is an asynchronous technology, which enables delivery to be retried until it succeeds. In contrast, most applications use synchronous function calls; for example: a procedure calling a sub-procedure, one method calling another method, or one procedure invoking another remotely through a remote procedure call (RPC) (such as CORBA and DCOM). Synchronous calls imply that the calling process is halted while the sub-process is executing a function. Even in an RPC scenario, where the called sub-procedure executes in a different process, the caller blocks until the sub-procedure returns control (and the results) to the caller. When using asynchronous messaging, the caller uses a *send and forget* approach that allows it to continue to execute after it sends the message. As a result, the calling procedure continues to run while the sub-procedure is being invoked.



Synchronous and Asynchronous Call Semantics

Asynchronous communication has a number of implications. First, we no longer have a single thread of execution. Multiple threads enable sub-procedures to run concurrently, which can greatly improve performance and help ensure that some sub-processes are making progress even while other sub-processes may be waiting for external results. However, concurrent threads can also make debugging much more difficult. Second, results (if any) arrive via a callback. This enables the caller to perform other tasks and be notified when the result is available, which can improve performance. However, the caller has to be able to process the result even while it is in the middle of other tasks, and it has to be able to use the result to remember the context in which the call was made. Third, asynchronous sub-processes can execute in any order. Again, this enables one sub-procedure to make progress even while another cannot. But it also means that the subprocesses must be able to run independently in any order, and the caller must be able to determine which result came from which sub-process and combine the results together. So asynchronous communication has several advantages but requires rethinking how a procedure uses its sub-procedures.

Source:

http://www.enterpriseintegrationpatterns.com/patterns/messaging/Introduction.html