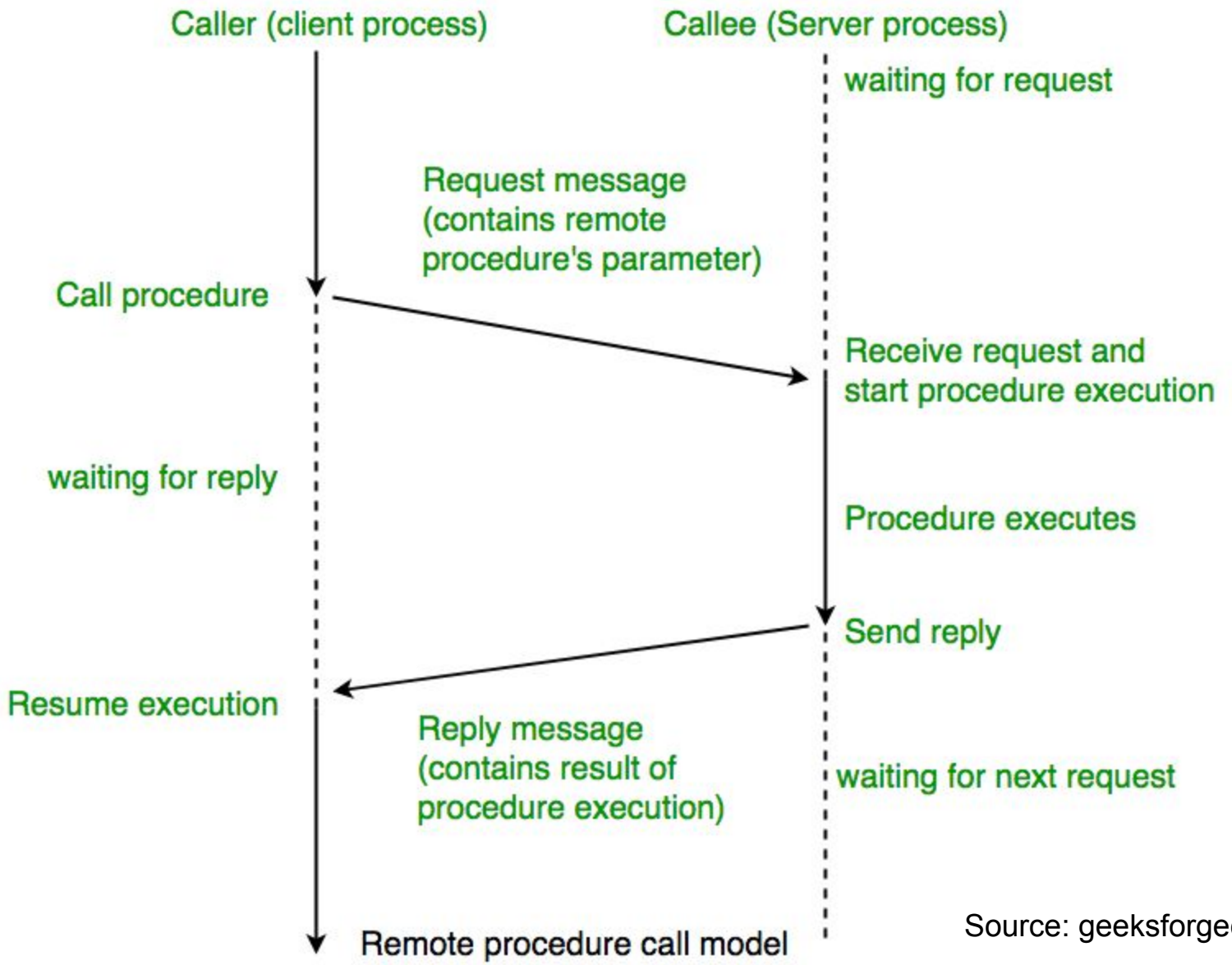
Remote Procedure Calls-RPC

It is an IPC mechanism used for communication between processes on different machines or processes on the same machine.

Remote Procedure Calls-RPC

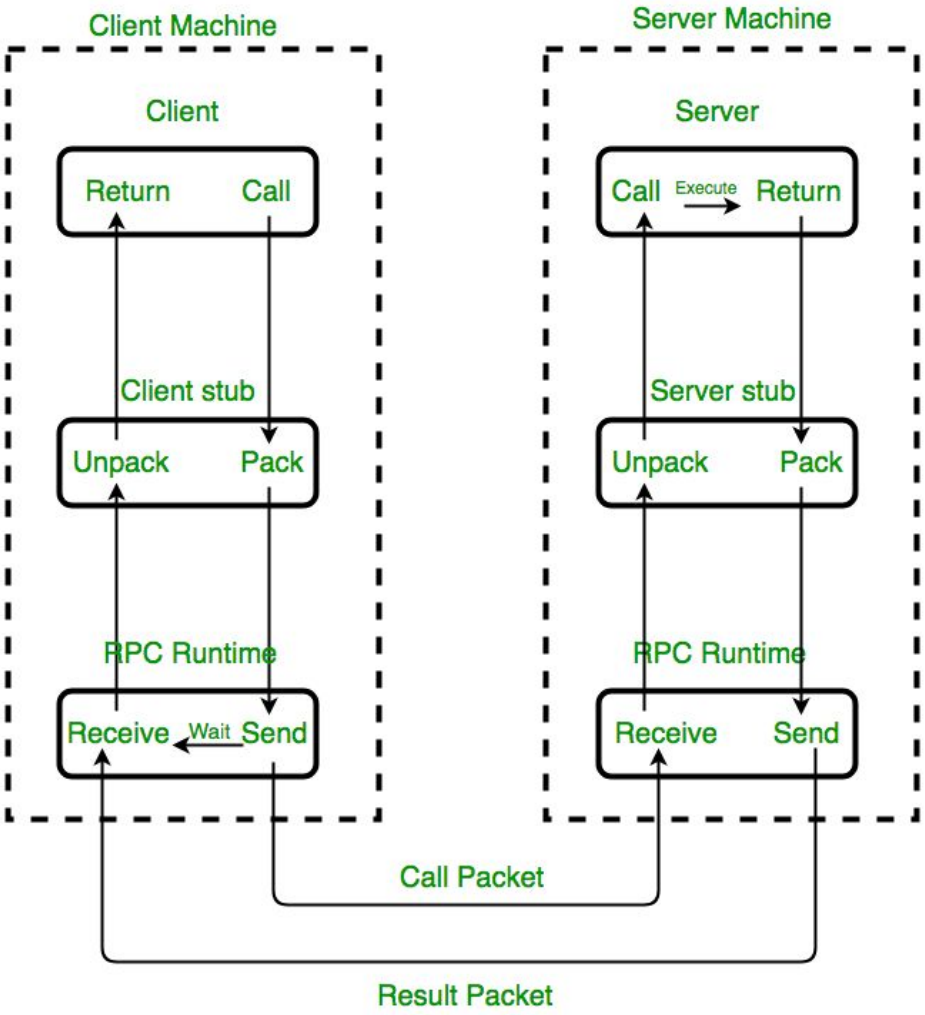
- For making a procedure call,the caller places arguments to the procedure in some well-specified location.
- Control is then transferred to the procedure
- The procedure executes in the newly created environment with the passed arguments
- After execution of the procedure the control returns to the calling point, possibly returning the result.

When making a Remote Procedure Call:



Source: geeksforgeeks

Remote Procedure Calls-RPC model



Implementation of RPC mechanism

Transparency of RPC

Syntactic Transparency: means that a remote procedure call should have exactly same syntax as LPC (local Procedure Call).

Semantic transparency : means that the semantics of a remote procedure call are identical to those of LPC.

Discussion of RPC and LPC

Implementing RPC mechanisms

Client : It initiates a remote procedure call

Client Stub :On receipt of a call request from client,it packs a specification of the target procedure and arguments into a message and asks RPC Runtime to send it to server stub.

On receipt of result of procedure execution,it unpacks the result and passes to the client.

RPC Runtime : handles transmission of messages across the network between client and server machines.

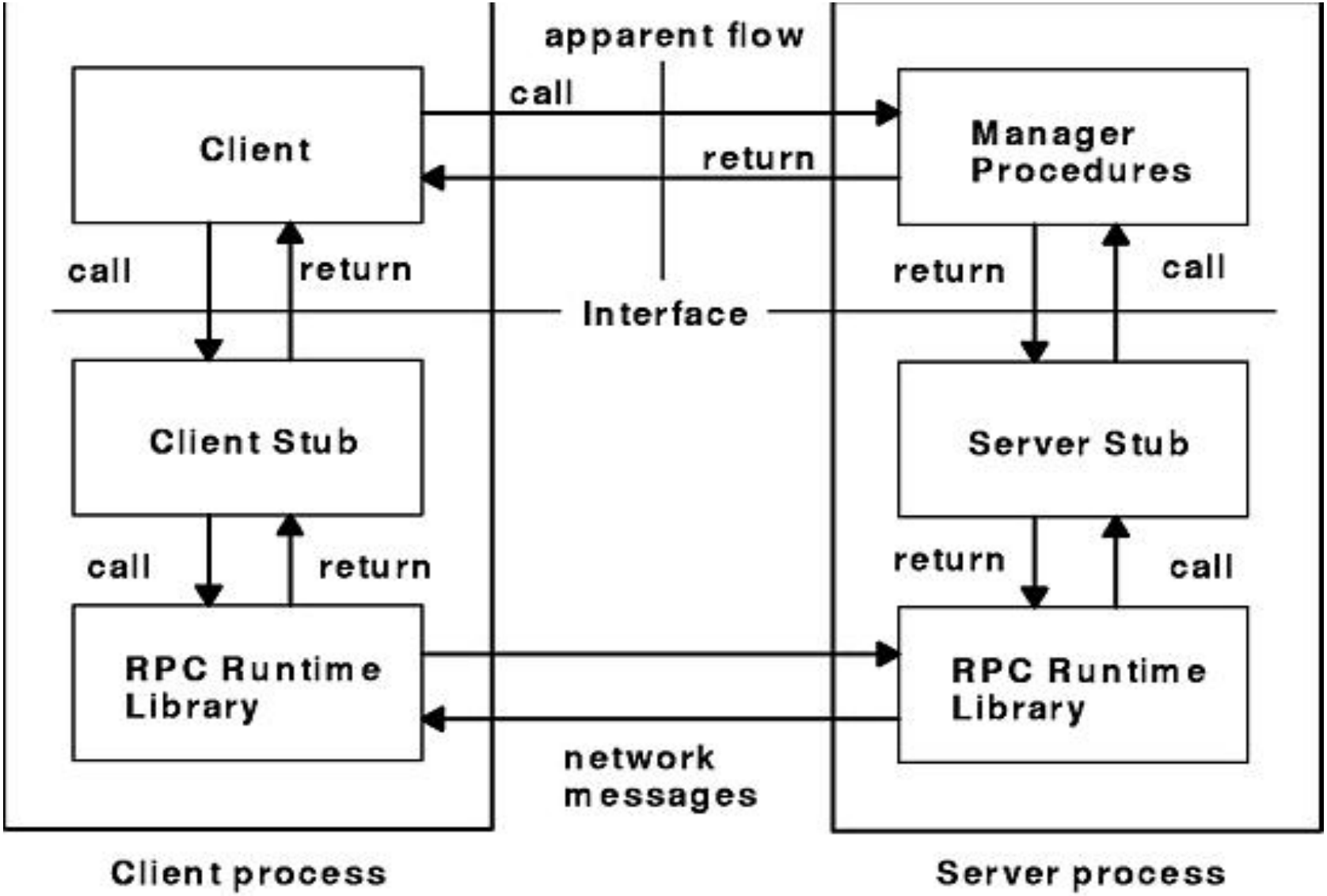
It is responsible for retransmissions, acknowledgements,packet routing, encryption.

Implementing RPC mechanisms

Server Stub : On receipt of the call request message from the Local RPC Runtime ,the server stub unpacks it and makes a perfectly normal call to invoke appropriate procedure of the server.

On receipt of the result of procedure execution from the server,the server stub packs the result into a message and then asks the local RPC Runtime to send it to client stub

Server : On receiving a call request from server stub,the server executes the appropriate procedure and returns result of procedure execution to server stub.



Remote Procedure Call Flow

Stub Generation

Manually : The RPC implementor provides a set of translation functions from which a user can construct his/her own stubs

Simple to implement,and can handle complex parameters.

Automatically : Commonly used method. It uses IDL :Interface definition language.

IDL : Interface Definition Language

IDL : Used to define interface between client and server.

It is a list of procedure names supported by the Interface,together with types of their arguments and results.

Client and Server independently perform compile time type checking.

IDL can indicate whether each argument is i/p,o/p or both.

IDL has information to type definitions,constants etc

The server is said to EXPORT the interface and the client is said to IMPORT the interface.

In a distributed application the programmer writes the interface using IDL

IDL : Interface Definition Language

IDL : Used to define interface between client and server.

It is a list of procedure names supported by the Interface,together with types of their arguments and results.

Client and Server independently perform compile time type checking.

IDL can indicate whether each argument is i/p,o/p or both.

IDL has information to type definitions,constants etc

The server is said to EXPORT the intetrface and the client is said to IMPORT the interface.

RPC messages

Call messages from the client to server

Reply messages from the server to the client.

RPC Call message format :

1. **Message ID**
2. **Message Type**
3. **Client Identifier**
4. **Remote procedure Identifier**
   - **program number**
   - **Version number**
   - **Procedure number**
5. **Arguments**

RPC messages

RPC reply message format :

Message Identifier
Message Type
Reply Status (success/failure)
Result

Server Management

Stateful servers / Stateless servers

Server creation semantics :-

- **Instance per call servers :-**

  Exist for duration of a single call only

  These servers are stateless.

- **Instance per session servers**

  Exist for an entire session

  Can be stateful

  Persistent Servers

  Exist indefinitely

  Can be bound to several clients

  Stateful

RPC parameter passing and Call semantics

- **Call by value**

The client stub copies and packages the value from the client into a message so that it can be sent to the server through a network.

- Parameter marshaling is the process of wrapping up parameters into a message. Consider the remote procedure add(x, y), which returns the addition of two integer parameters.
- The client stub stores its two parameters in a message, along with the name or number of the called method.
- The stub evaluates the message when it arrives at the server to determine which method is required, and then performs the necessary call.
- The client puts the server's result into a message after the execution has been completed by the server. Then this message is received by the client stub and its unpacking is carried out before returning the value to the client procedure.
- This parameter parsing semantic works well as long as the client and server computers are similar and all parameters and outcomes are scalar types like Booleans, characters, and integers.

RPC parameter passing and Call semantics

- **Call by reference**
- Parameters can be passed by reference in distributed systems with distributed shared memory techniques, for example.
- If on the client-side, the second parameter is the buffer's address, suppose 100 then one cannot simply provide 100 to the server and expect it to function as address 100 may be in the middle of the program text on the server.
- In some systems, it is managed by sending the pointer to the server stub and implementing special pointer-specific logic in the server function.
- A pointer can be followed by saving it in a register and then following it indirectly through the register (dereferenced).
- When this technique is used, a pointer is dereferenced by sending a message back to the client stub telling it to fetch and deliver the item pointed to (reads) or save a value at the address pointed to (writes).
- While this method is effective, it is inefficient most of the time.

RPC call semantics

“maybe” call semantics

- no retransmission of request messages
- not certain whether the procedure has been executed
- no fault-tolerance measures
- generally not acceptable

RPC parameter passing and Call semantics

“at-least-once” call semantics

– It guarantees call is executed one/more times but does not specify which results are returned to the caller.

RPC parameter passing and Call semantics

“last one call” semantics

Based on retransmission of call message based on timeouts

Procedure is executed many times.

Last call results are used by the caller.

RPC parameter passing and Call semantics

“exactly once call semantics”

This is the strongest semantic

Eliminates procedure being executed more than once no matter how many times the call is retransmitted.

Problems in RPC

The request is lost

The reply is lost

The client and server may crash

Communication protocols for RPC

The request Protocol

The request /reply Protocol

The request/reply/acknowledgement Protocol

Client-Server binding in RPC

Binding : refers to *determining the location and identity* (communication ID) of the called procedure.

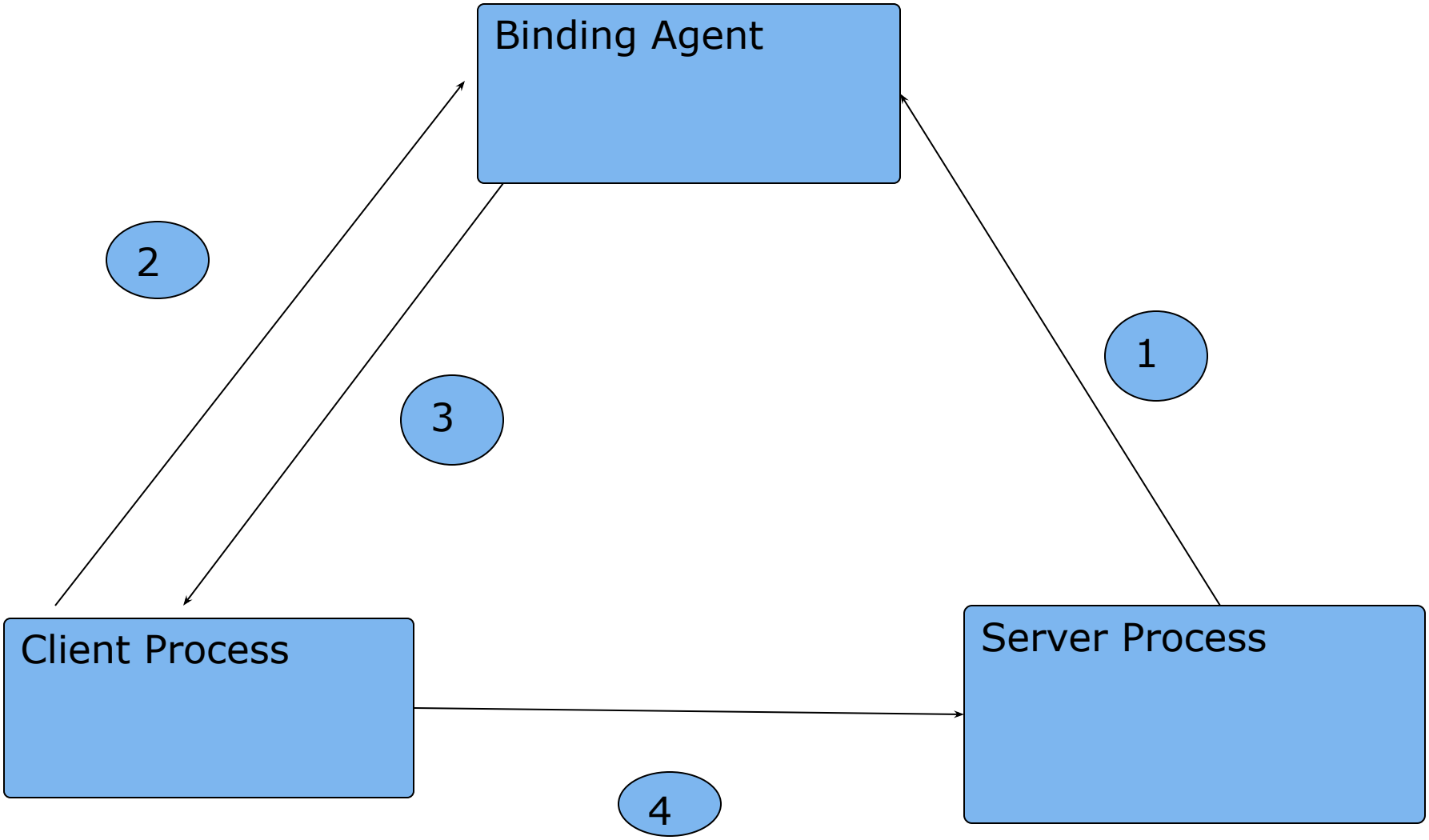
static binding : (which binds the host address of a server into the client program at compilation time),less desirable

- The client and server programs are compiled separately and often at different times
- The server may be moved from one host to another

dynamic binding : (which binds the host address of a server into the client program at run time)

- allows servers to register their exporting services
- allows servers to remove services
- allows clients to lookup the named service

Locating a server in case of RPC



Binding Time

Binding at Compile Time :

Hard-coding the servers network address into client code

-ve : if server moves to a different location,all client programs will have to be recompiled.

Binding at Link Time :

The server exports is services to the BA(Binding Agent)

The client makes an import before making a call.

The BA returns the server handle to the the client.

The client connects to the server directly via the handle

Binding Time

Binding at call time :

The client is bound to the server when it calls the server for the first time.

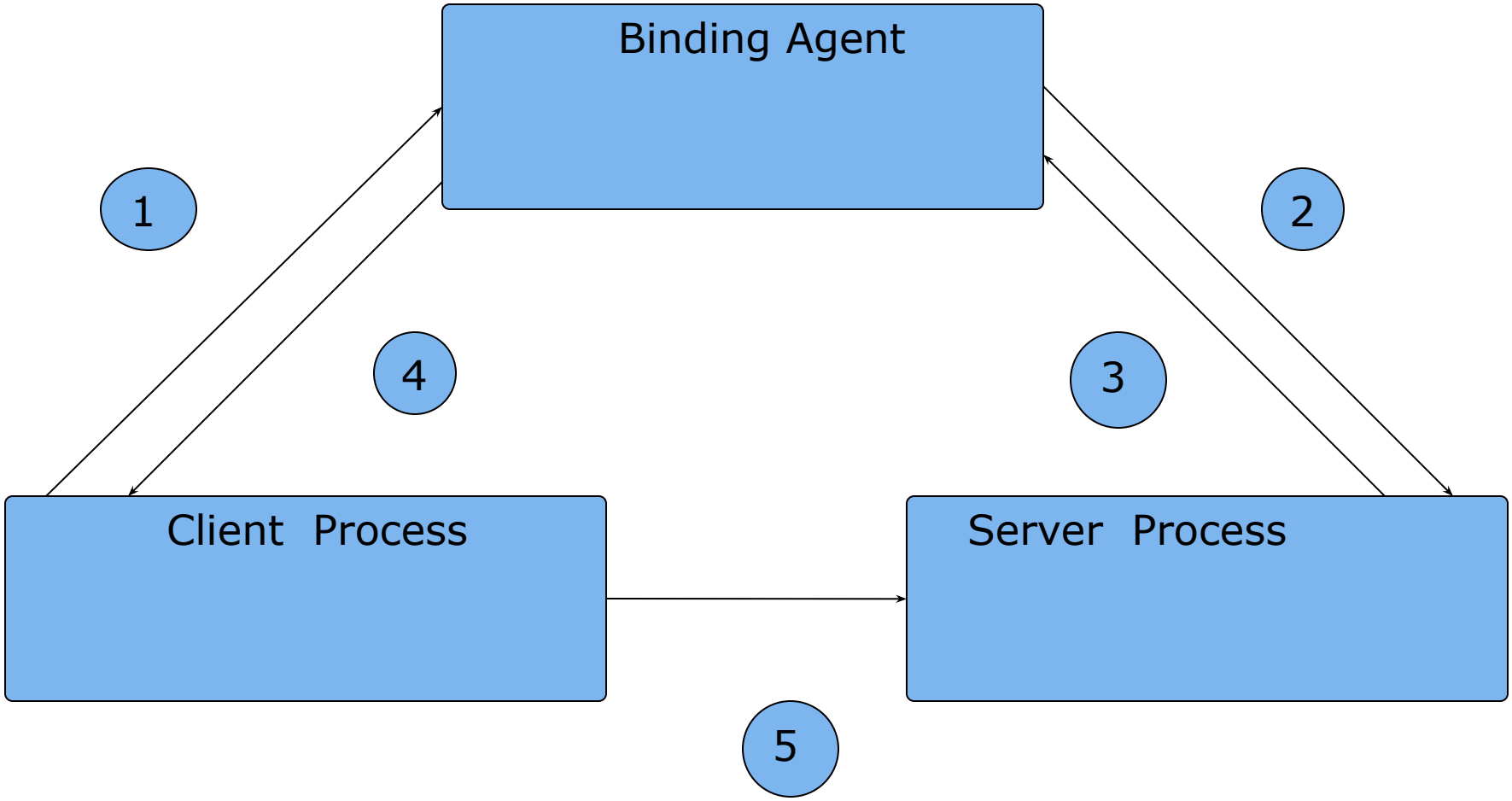
Client passes the server's interface name and arguments of RPC call to the binding agent.

The binding agent looks up the location of target server and on behalf of client sends RPC to target server.

The target server returns the result to BA and it returns the result to the Client

Later calls are sent directly to the server

Client-Server call time binding in RPC



Special types of RPCs

Call back RPC

It allows peer to peer paradigm

Client makes a call to the server and during execution the server calls back the client for certain details.

Broad cast RPC mechanisms

The client broadcasts the messages to the binding agent and the binding agent then sends them to the servers

Or the clients can also send messages to the broadcast port.

Batch mode RPC

Is to queue separate RPC requests in a transmission buffer on client side and then send them in one batch to the server.

Overview of RPC Systems

Sun RPC

DCE RPC

DCOM

CORBA

Java RMI

XML RPC, SOAP/.NET, AJAX, REST

Sun RPC

RPC for Unix System V, Linux, BSD

Also known as ONC RPC
(Open Network Computing)

Interfaces defined in an Interface Definition Language (IDL)

– IDL compiler is **rpcgen**

Programming with RPCs

1. A .x file
2. Client.c
3. Server.c

4. Header files need to be added
5. Client and server stubs are auto generated

https://tharikasblogs.blogspot.com/p/how-to-write-simple-rpc-programme.html

Programming with RPCs