Remote Procedure Calls

Introduction

- IPC part of distributed system can often be conveniently handled by message-passing model.
- It doesn't offer a uniform panacea for all the needs.
- RPC emerged as a result of this.
- It can be said as the special case of message-passing model.

Cont…

- It has become widely accepted because of the following features:
  - Simple call syntax and similarity to local procedure calls.
  - It specifies a well defined interface and this property supports compile-time type checking and automated interface generation.
  - Its ease of use, efficiency and generality.
  - It can be used as an IPC mechanism between
    - processes on different machines and
    - also between different processes on the same machine.

RPC Model

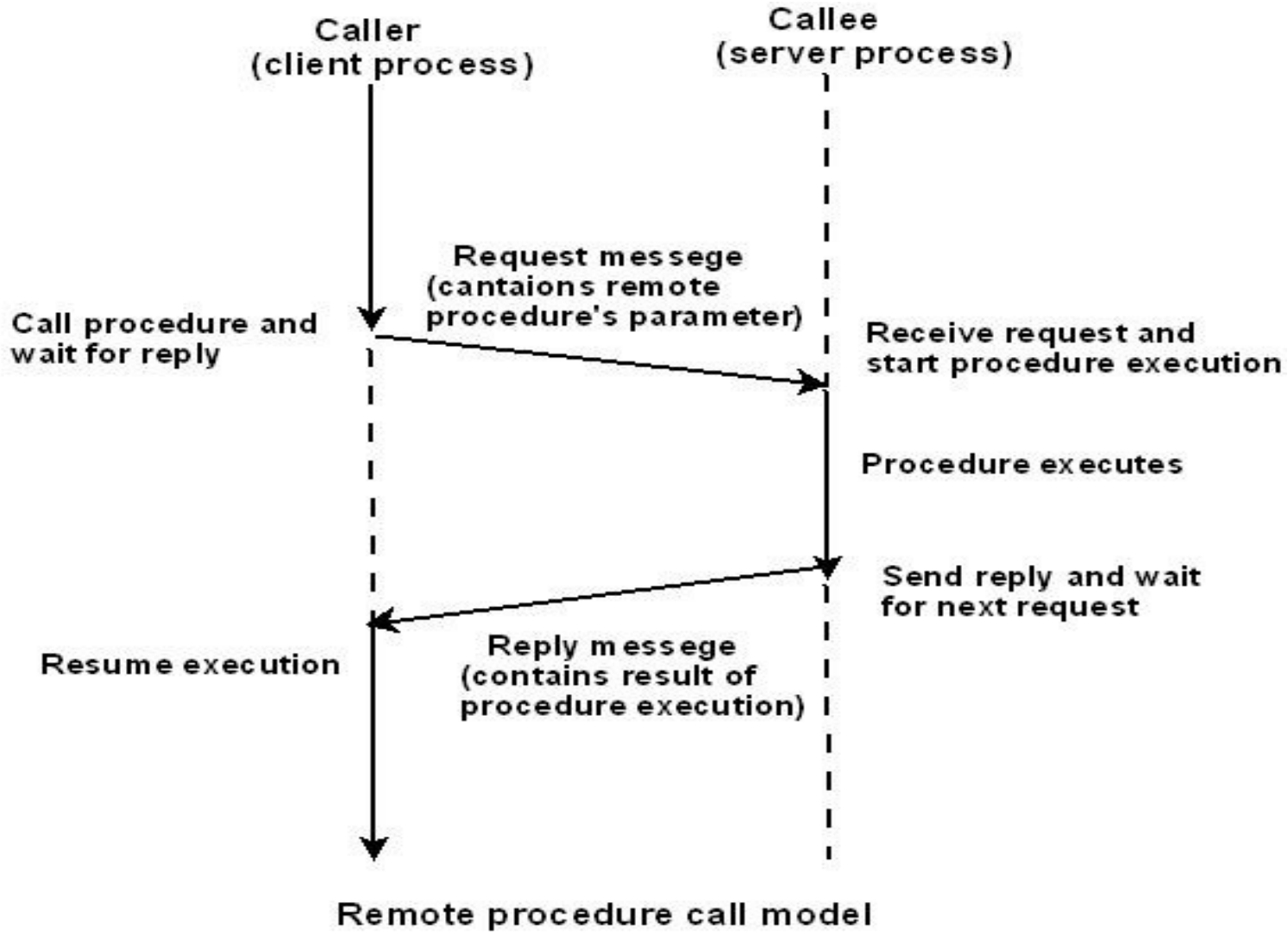
- It is similar to commonly used procedure call model. It works in the following manner:

1. For making a procedure call, the caller places arguments to the procedure in some well specified location.
2. Control is then transferred to the sequence of instructions that constitutes the body of the procedure.
3. The procedure body is executed in a newly created execution environment that includes copies of the arguments given in the calling instruction.
4. After the procedure execution is over, control returns to the calling point, returning a result.

Cont…

- The RPC enables a call to be made to a procedure that does not reside in the address space of the calling process.
- Since the caller and the callee processes have disjoint address space, the remote procedure has no access to data and variables of the callers environment.
- RPC facility uses a message-passing scheme for information exchange between the caller and the callee processes.
- On arrival of request message, the server process
  - extracts the procedure's parameters,
  - computes the result,
  - sends a reply message, and
  - then awaits the next call message.

Cont…



Remote procedure call model

Cont…

- Only one of the two processes is active at any given time.
- It is not always necessary that the caller gets blocked.
- There can be RPC implementations depending on the parallelism of the caller and the callee's environment.
- The RPC could be asynchronous, so that the client may do useful work while waiting for the reply from the server.
- Server could create a thread to process an incoming request so that server is free to receive other requests.

Transparency of RPC

- A transparent RPC is one in which the local and remote procedure calls are indistinguishable.
- Types of transparencies:
  - *Syntactic transparency*
    - A remote procedure call should have exactly the same syntax as a local procedure call.
  - *Semantic transparency*
    - The semantics of a remote procedure call are identical to those of a local procedure call.
- Syntactic transparency is not an issue but semantic transparency is difficult.

Cont…

- Difference between remote procedure calls and local procedure calls:

1. Unlike local procedure calls, with remote procedure calls,
   - Disjoint Address Space
   - Absence of shared memory.
   - Meaningless making call by reference, using addresses in arguments and pointers.
2. RPC's are more vulnerable to failure because of:
   - Possibility of processor crashes or
   - communication problems of a network.

Cont…

3. RPC's are much more time consuming than LPC's due to the involvement of communication network.

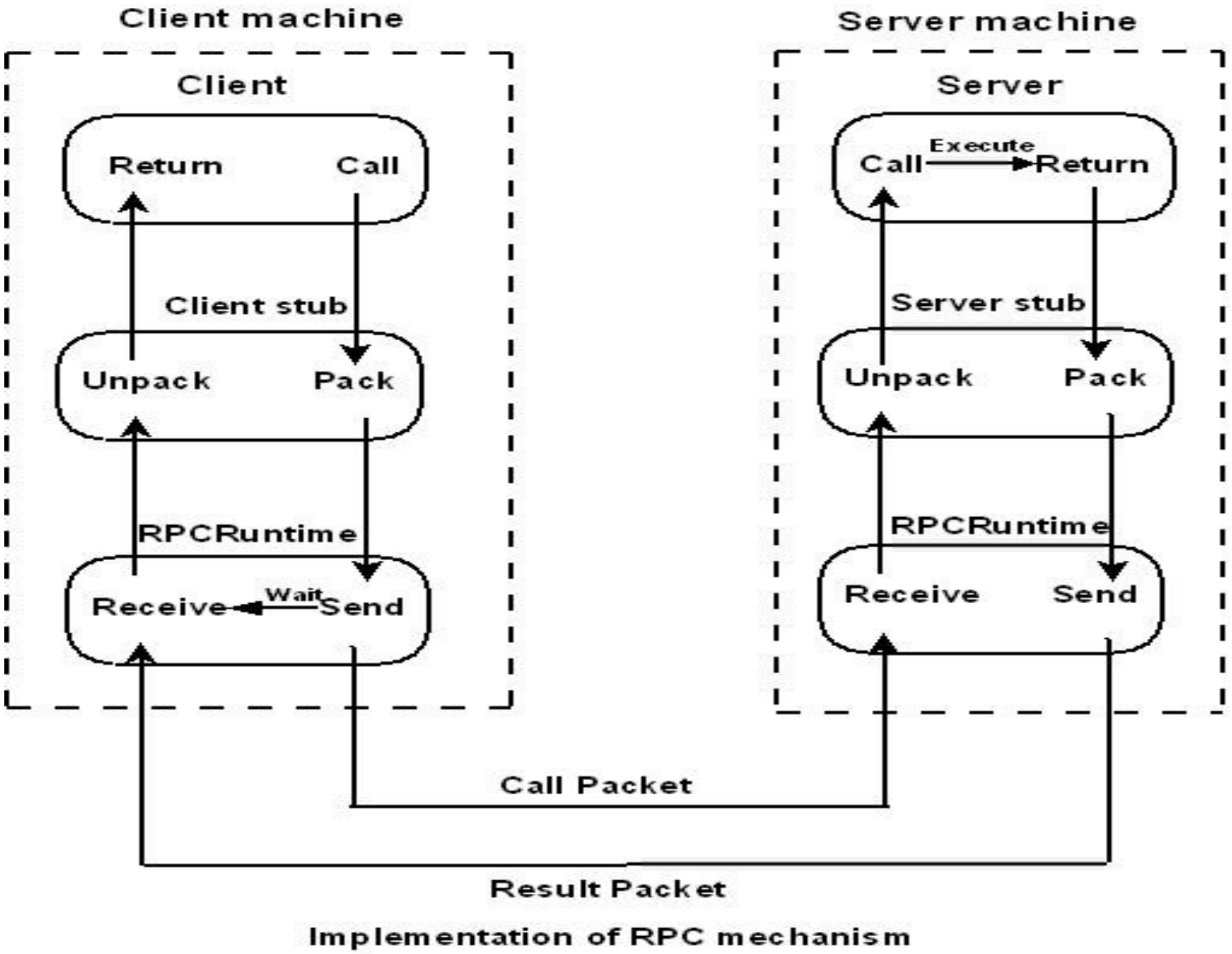
- Due to these reasons, total semantic transparency is impossible.

Implementing RPC Mechanism

- To achieve semantic transparency, implementation of RPC mechanism is based on the concepts of stubs.
- Stubs
  - Provide a normal / local procedure call abstraction by concealing the underlying RPC mechanism.
  - A separate stub procedure is associated with both the client and server processes.
  - To hide the underlying communication network, RPC communication package known as RPC Runtime is used on both the sides.

Cont…

- Thus implementation of RPC involves the five elements of program:
  1. Client
  2. Client Stub
  3. RPC Runtime
  4. Server stub
  5. Server
- The client, the client stub, and one instance of RPCRuntime execute on the client machine.
- The server, the server stub, and one instance of RPCRuntime execute on the server machine.
- As far as the client is concerned, remote services are accessed by the user by making ordinary LPC



Implementation of RPC mechanism

Client Stub

- It is responsible for the following two tasks:
  - On receipt of a call request from the client,
    - it packs a specifications of the target procedure and the arguments into a message and
    - asks the local RPC Runtime to send it to the server stub.
  - On receipt of the result of procedure execution, it unpacks the result and passes it to the client.

RPCRuntime

- It handles transmission of messages across the network between Client and the server machine.

- It is responsible for
  - Retransmission,
  - Acknowledgement,
  - Routing and
  - Encryption.

Server Stub

- It is responsible for the following two tasks:
  - On receipt of a call request message from the local RPCRuntime, it unpacks it and makes a perfectly normal call to invoke the appropriate procedure in the server.
  - On receipt of the result of procedure execution from the server, it unpacks the result into a message and then asks the local RPCRuntime to send it to the client stub.

Stub Generation

- Stubs can be generated in one of the following two ways:
  - Manually
  - Automatically

Automatic Stub Generation

- Interface Definition Language (IDL) is used here, to define the interface between a client and the server.

- Interface definition:
  - It is a list of procedure names supported by the interface together with the types of their arguments and results.
  - It also plays role in reducing data storage and controlling amount of data transferred over the network.
  - It has information about type definitions, enumerated types, and defined constants.

Cont…

- Export the interface
  - A server program that implements procedures in the interface.
- Import the interface
  - A client program that calls procedures from an interface.
- The interface definition is compiled by the IDL compiler.
- IDL compiler generates
  - components that can be combined with client and server programs, without making any changes to the existing compilers;
  - Client stub and server stub procedures;
  - The appropriate marshaling and unmarshaling operations;
  - A header file that supports the data types.

RPC Messages

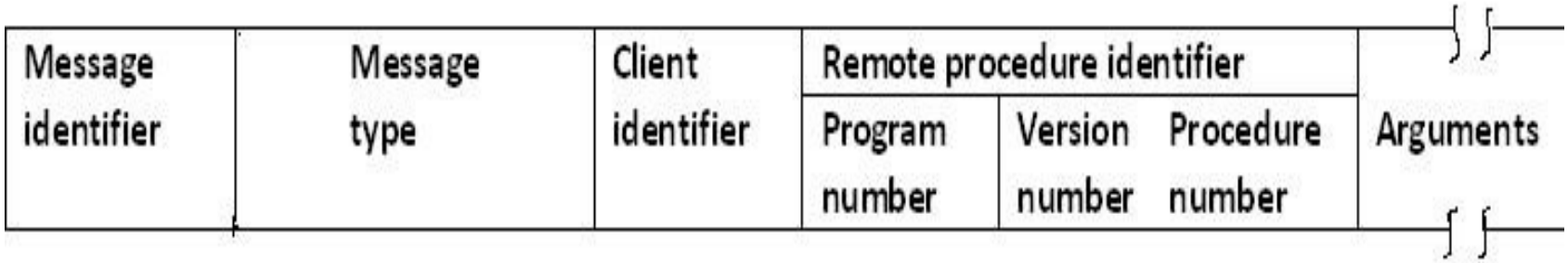
- RPC system is independent of transport protocols and is not concerned as to how a message is passed from one process to another.

- Types of messages involved in the implementation of RPC system:
  - Call messages
  - Reply messages

Call Messages

- Components necessary in a call message are
  1. The id. Information of the remote procedure to be executed.
  2. The arguments necessary for the execution.
  3. Message Identification field consists of a sequence number for identifying lost and duplicate messages.
  4. Message Type field: whether call or reply messages.
  5. Client Identification Field allows the server to:
     - Identify the client to whom the reply message has to be returned and
     - To allow server to authenticate the client process.

Call Message Format

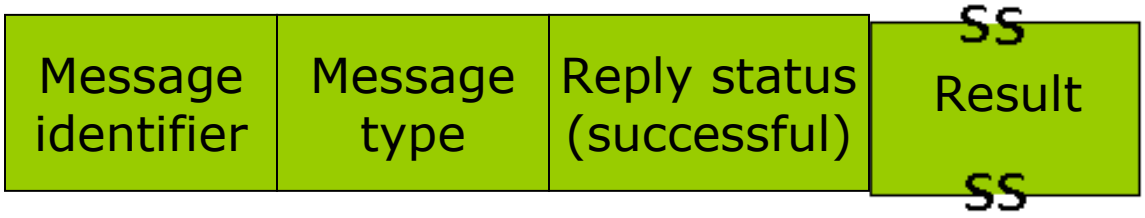


A typical RPC call message format

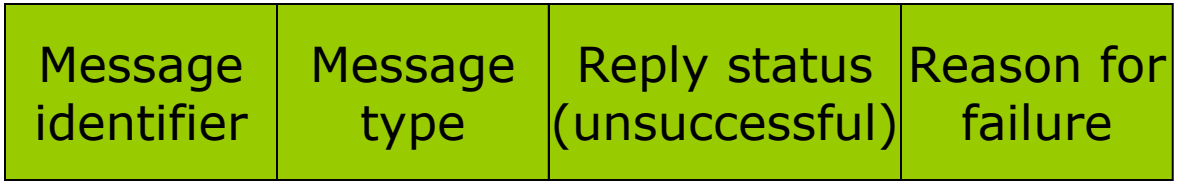
Reply Messages

- Sent by the server to the client for returning the result of remote procedure execution.
- Conditions for unsuccessful message sent by the server:
  - The server finds that the call message is not intelligible to it.
  - Client is not authorized to use the service.
  - Remote procedure identifier is missing.
  - The remote procedure is not able to decode the supplied arguments.
  - Occurrence of exception condition.

Reply message formats



A successful reply message format



An unsuccessful reply message format

Marshalling Arguments and Results

- The arguments and results in RPC are language-level data structures, which are transferred in the form of message data.
- Transfer of data between two computers requires encoding and decoding of the message data.
- Encoding and decoding of messages in RPC is known as marshaling & Unmarshaling.
- Classification of marshaling procedures:
  - Those provided as a part of the RPC software.
  - Those that are defined by the users of the RPC system.

Server Management

- Issues :
  - Server implementation
  - Server creation

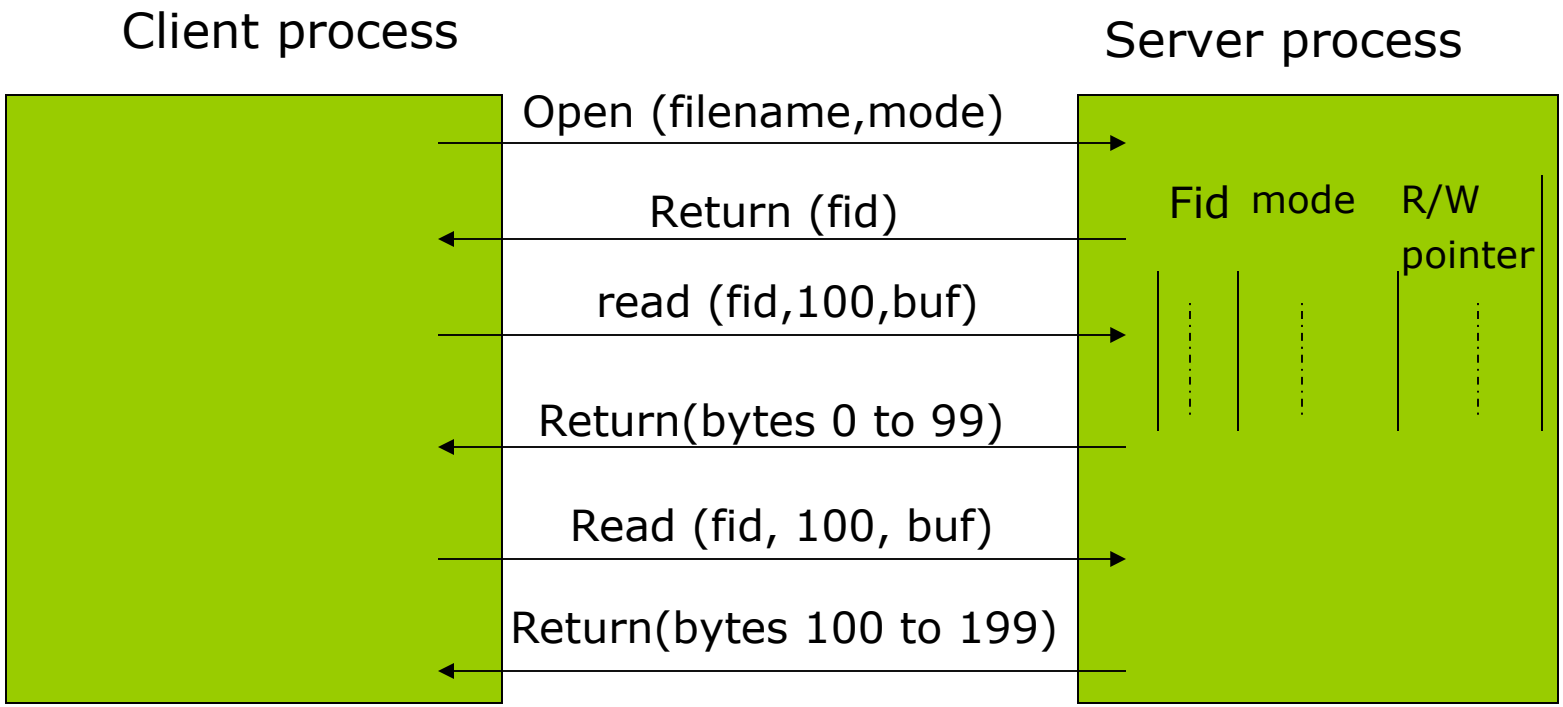
Server Implementation

- Types of servers :
  - Stateful servers
  - Stateless servers

Stateful Server

- A Stateful Server maintains clients state information from one RPC to the next.
- They provide an easier programming paradigm.
- They are more efficient than stateless servers.

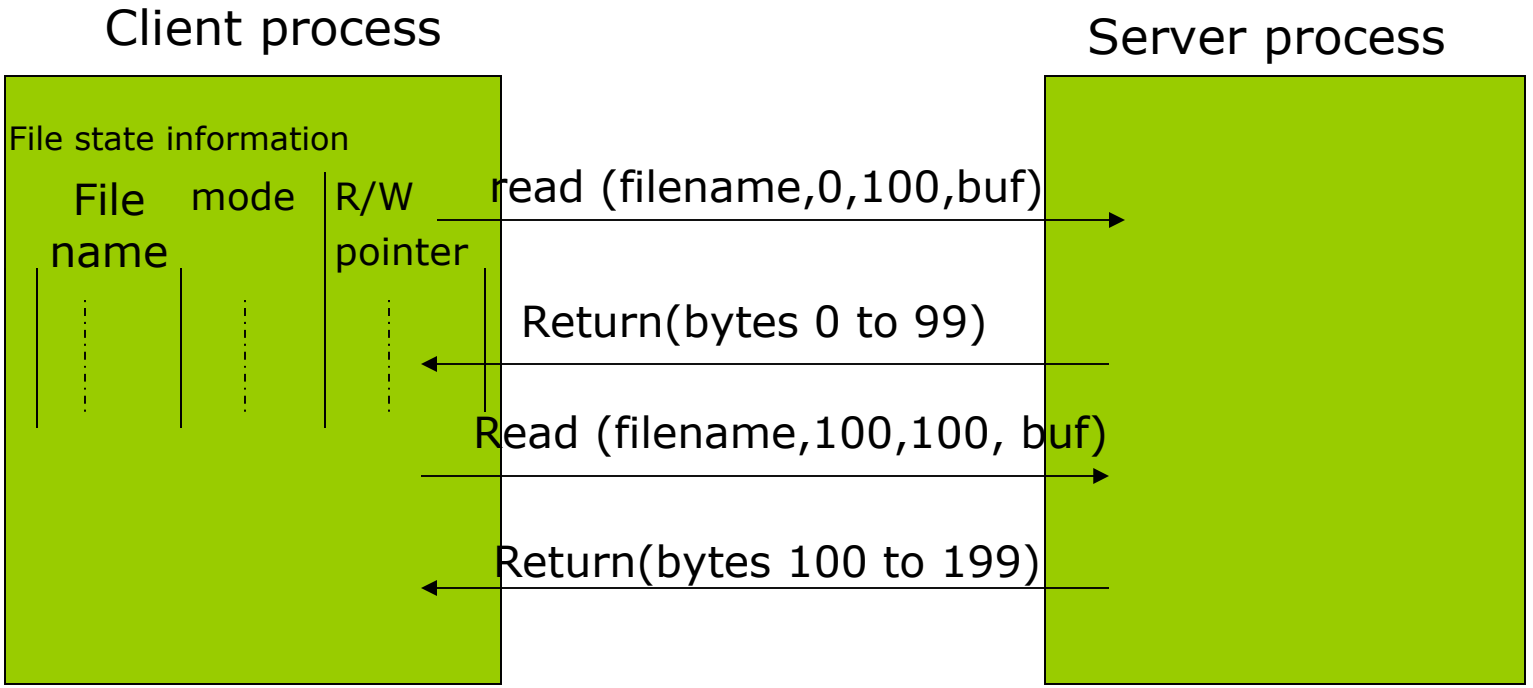
Stateful file server



Stateless Server

- Every request from a client must be accompanied with all necessary parameters to successfully carry out the desired operation.
- Stateless servers have distinct advantage over stateful server in the event of a failure.
- The choice of using a stateless or a stateful server is purely application dependent.

Stateless file server



Server Creation Semantics

- A server process is independent of a client process that makes a remote procedure call to it.
- Server processes may either be created and installed before their client processes or be created on a demand basis.
- Based on the life duration for which RPC servers survive:
  - Instance-per-call Servers
  - Instance-per-session Server
  - Persistent Servers

Instance-per-call Servers

- They exist only for the duration of a single call.
- These server are created by RPCRuntime on the server machine only when a call message arrives.
- The server is deleted after the call has been executed.
- The servers of this type are
  - Stateless, and
  - more expensive for invoking same type of service.

Instance-per-session Server

- Exist for entire session for which C & S interact.
- It can maintain intercall state information to minimize
  - the overhead involved in server creation and
  - destruction for a client-server session.
- Server managers are used for each type of services.
- Server managers are registered with the binding agent.
- A server of this type only services a single client and hence only has to manage a single set of state information.

Persistent Servers

- Persistent servers remain in existence indefinitely.
- Advantages:
  - Most commonly used.
  - Improves performance and reliability.
  - Shared by many clients.
- Each server exports its services by registering with binding agent.

Parameter Passing Semantics

- Call by Value Semantics
- Call by Reference Semantics
  - Call-by-object-reference
  - Call-by-move

Call-by-value

- In the call-by-value method, all parameters are copied into a message that is transmitted from the client to the server through the network.
- It is time consuming for passing large data types such as trees, arrays, etc.

Call-by-reference

- Most RPC mechanisms use the call-by-reference semantics for parameter passing
  - The client and the server exist in different address space.
- Distributed systems having distributed shared-memory mechanisms can allow passing of parameters by reference.
- Call-by-object-reference
  - objects invocation is used by the RPC mechanism.
  - Here, the value of a variable is a reference to an object.

Call-by-reference

- Call-by-move
  - A parameter is passed by reference as in the method of call-by-object-reference, but at the time of the call,
  - the parameter object is moved to the destination node (callee).
  - If it remains at the caller's node ,it is known as **call-by-visit**.
  - It allows packaging of the argument objects in the same network packet as the invocation message, thereby
    - reducing the network traffic and
    - message count.

Call Semantics

- Failure of communication link between the caller and the callee node is possible.

- Failures can be because of
  - Call Message
  - Reply
  - Caller / Callee crash or
  - Link

Failure Handling code is part of RPC Runtime

Types of Call Semantics

- Possibly or May-Be Call Semantics
- Last-one call semantics
- Last-of-many call semantics
- At-least-once call semantics
- Exactly-once call semantics

Possibly or May-Be Call Semantics

- It is a Request Protocol.
- It uses time out but no surety of reply.
- It is an Asynchronous RPC.
- Server need not send back the reply.
- Useful for periodic update services.

Last-One Call Semantic

- It is Request / Reply protocol.
- Retransmission based on timeout.
- After time out, result of last execution is used by the caller.
- It issues Orphan Call (crash of caller).
- It is useful in designing simple RPC.

Last-of-many Call Semantic

- Similar to last-one semantic except that the orphan calls are neglected using call identifiers.
- When a call is repeated, it is assigned a new call identifier.
- A caller accepts a response only if the call identifier associated with it matches with the identifier of the most recently repeated call.

At-least-once Call Semantic

- It guarantees that the call is executed one or more times but does not specify which results are returned to the caller.
- It uses timeout-based retransmissions without caring for the orphan calls.
- For nested calls
  - If there are any orphan calls, it takes the result of the first response message and
  - ignores the others, whether or not the accepted response is from an orphan.

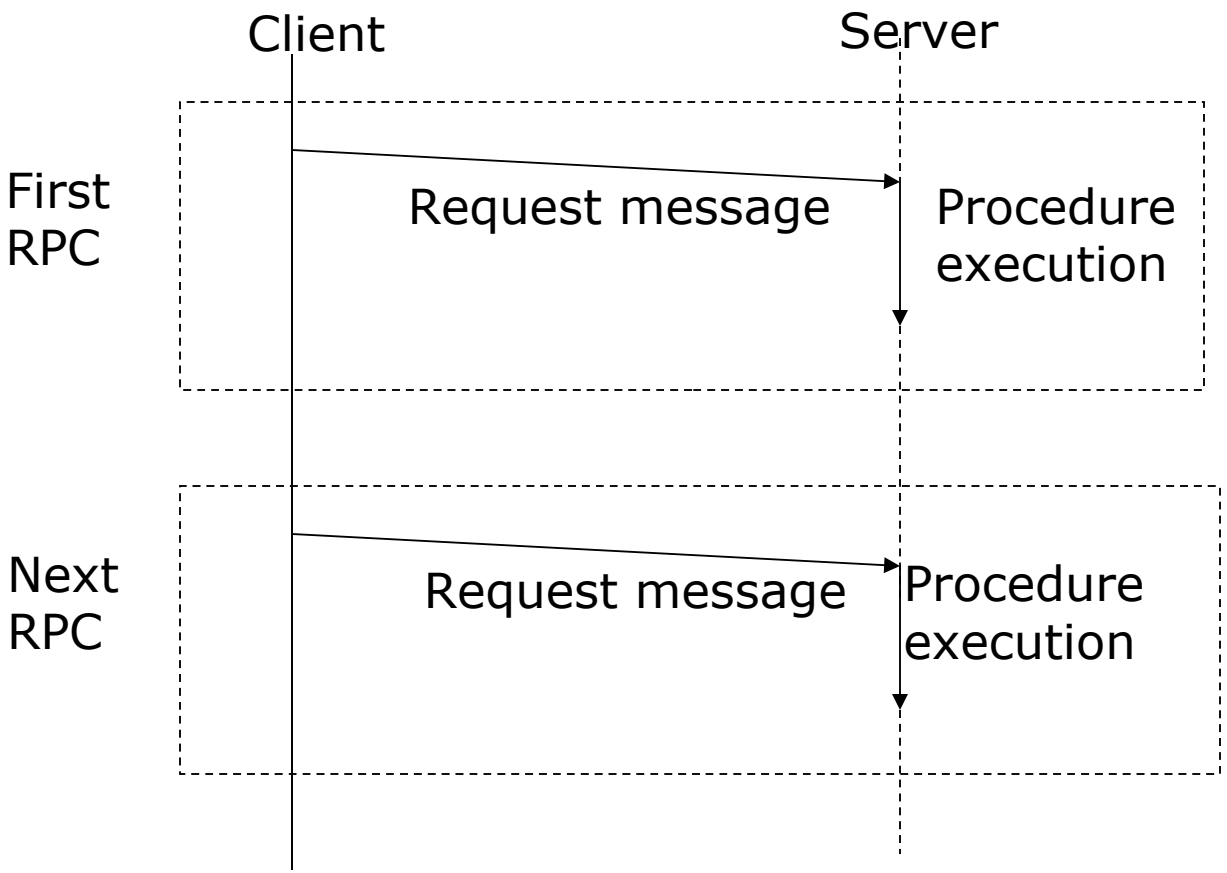
Exactly-Once Call Semantics

- It is Request / Reply / ACK protocol.
- No matter how many calls are made, a procedure is executed only once.
- The server deletes an information from its reply cache only after receiving an ACK from the client.

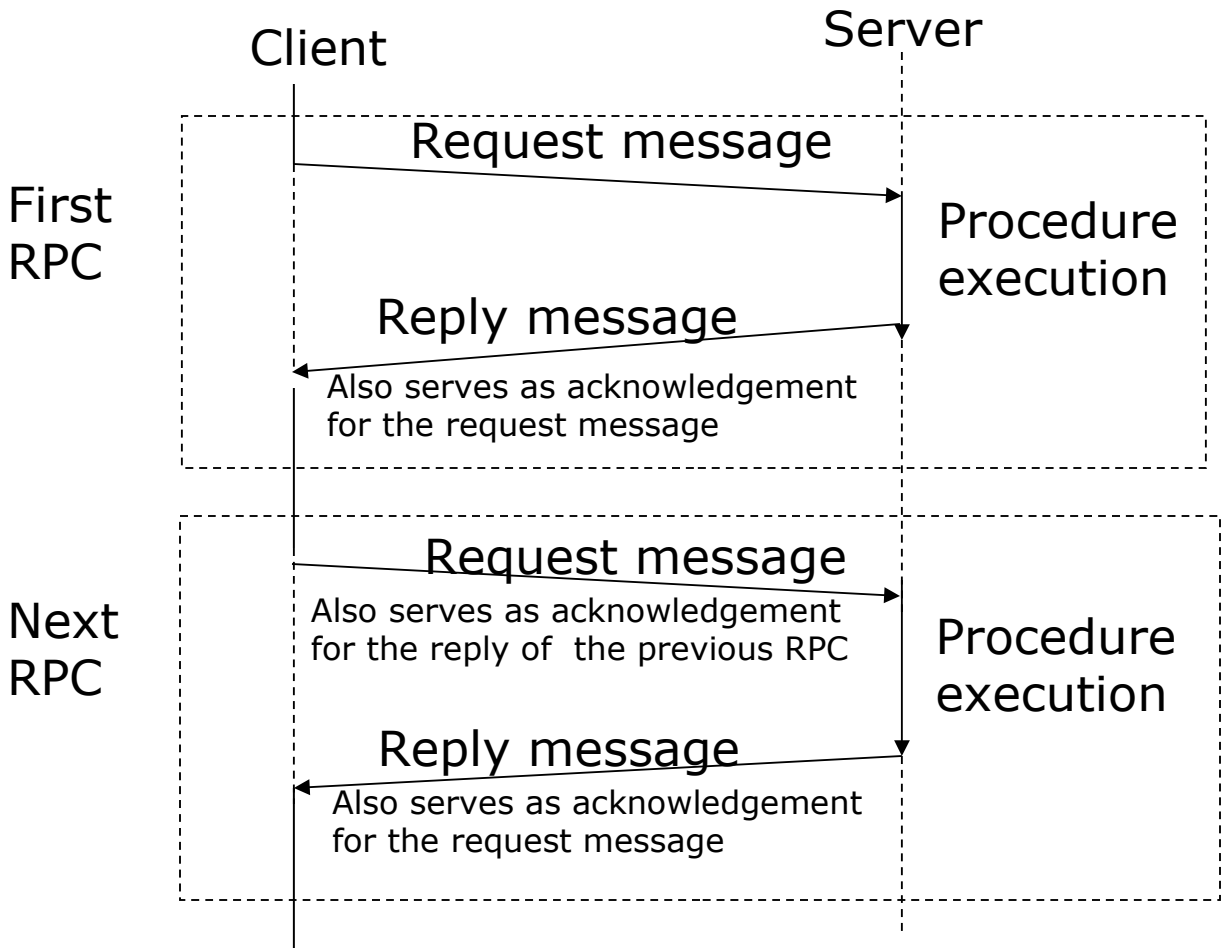
Communication protocols for RPCs

1. Request protocol
2. Request / Reply protocol
3. Request /Reply /Acknowledge-Reply protocol

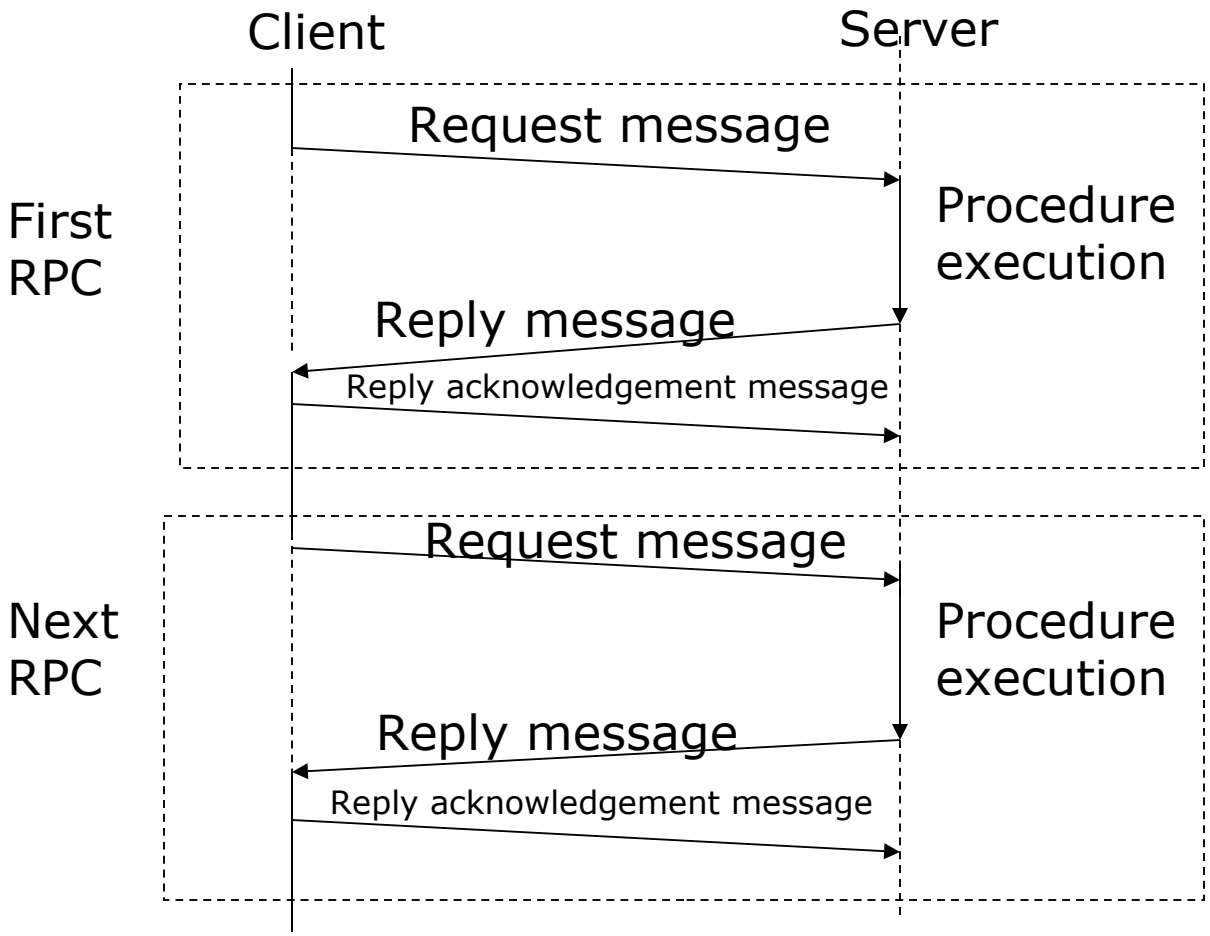
Request (R) protocol



Request / Reply (RR) protocol



Request / Reply / Ack. (RRA) protocol



Complicated RPC's

- Types of complicated RPCs
  - RPC's involving long duration calls or Large gaps between calls
  - RPC's involving arguments and/or results that are too large to fit in a single datagram packet.

RPCs involving Long-Duration Calls & Large Gaps Between Calls

- Two methods to handle these RPCs:
  - Periodic probing of the server by the clients.
    - After client sends a request message, it periodically sends a probe packet to the server, which the server is expected to acknowledge.
    - This allows the client to detect a server's crash or communication link failures and to notify the corresponding user of an exception condition.
  - Periodic generation of an ACK by the server.
    - If a server is not able to generate the next packet significantly sooner than the expected retransmission time, it spontaneously generates an acknowledgement.

RPC's involving Long Messages

- It proposes the use of multidatagram messages.
- A single ACK packet is used for all packets of this multidatagram message.
- Another crude method is to break one logical RPC into several physical RPC as in SUN Microsystem where RPC is limited to 8 kilo bytes.

Client – Server Binding

- Client Stub must know the location of a server before RPC can take place between them.
- Process by which client gets associated with server is known as BINDING.
- Servers export their operations to register their willingness to provide services and
- Clients import operations, asking the RPCRuntime to locate a server and establish any state that may be needed at each end.

Cont…

- Issues for client-server binding process:
  - How does a client specify a server to which it wants to get bound?
  - How does the binding process locate the specified server?
  - When is it proper to bind a client to a server?
  - Is it possible for a client to change a binding during execution?
  - Can a client be simultaneously bound to multiple servers that provide the same service?

Server Naming

- The naming issue is the specification by a client of a server with which it wants to communicate.
- Interface name of a server is its unique identifier.
- It is specified by *type & instance.*
  - Type specifies the version number of different interfaces
  - Instance specifies a server providing the services.
- Interface name semantics are based on arrangement between the exporter and importer.

Cont…

- Interface names are created by the users, not by the RPC package.
- The RPC package only dictates the means by which an importer uses the interface name to locate an exporter.

Server locating

- Methods are:
  1. Broadcasting
  2. Binding agent

Cont…

- Broadcasting
  - A message to locate a node is broadcast to all the nodes from the client node.
  - Node having desired server returns a response message. (could be replicated)
  - Suitable for small networks.

Cont…

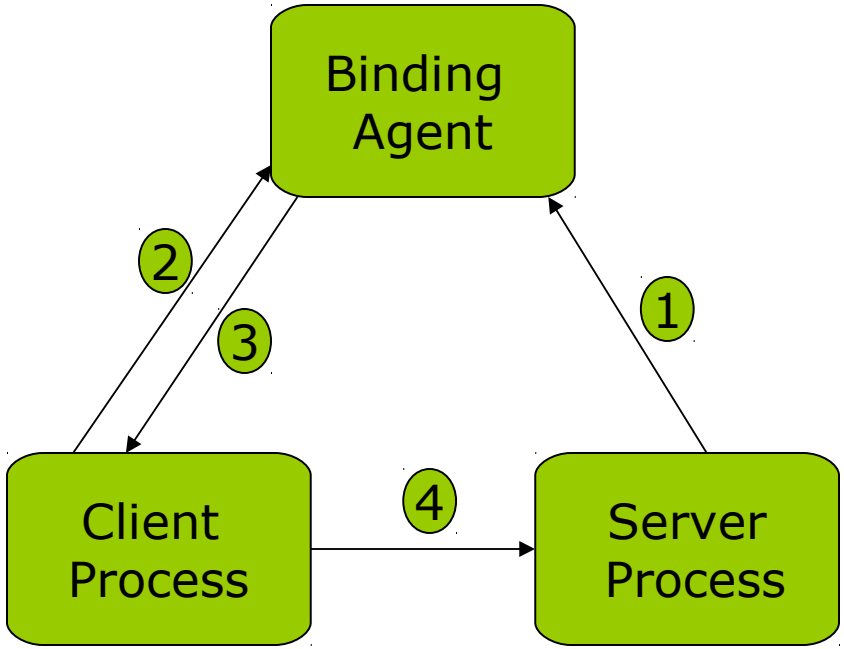
- Binding Agent
  - A name server used to bind a client to a server by providing the client with the location information.
  - Maintains a binding table containing mapping of server interface name to its location.
  - The table contains identification information and a handle to locate it.
  - The location of the binding agent (having a fixed address) is known to all nodes by using a broadcast message .

Cont…

- Primitives used by Binding Agent
  1. Register
  2. Deregister
  3. Lookup

Cont…

1. The server registers itself with the binding agent.
2. The client requests the binding agent for the server's location.
3. The binding agent returns the server's location information to the client.
4. The client calls the server.



Cont…

- Advantages
  - Support multiple servers having same interface type.
  - Client requests can be spread evenly to balance the load.
- Disadvantages
  - The overhead involved in binding clients to servers is large and becomes significant when many client processes are short lived.
  - A binding agent must be robust against failures and should not become a performance bottleneck.
- Solution
  - distribute binding function among several BA's, and
  - Replicate the information among them.
    - Again overhead in creating and handling replicas.

Binding Time

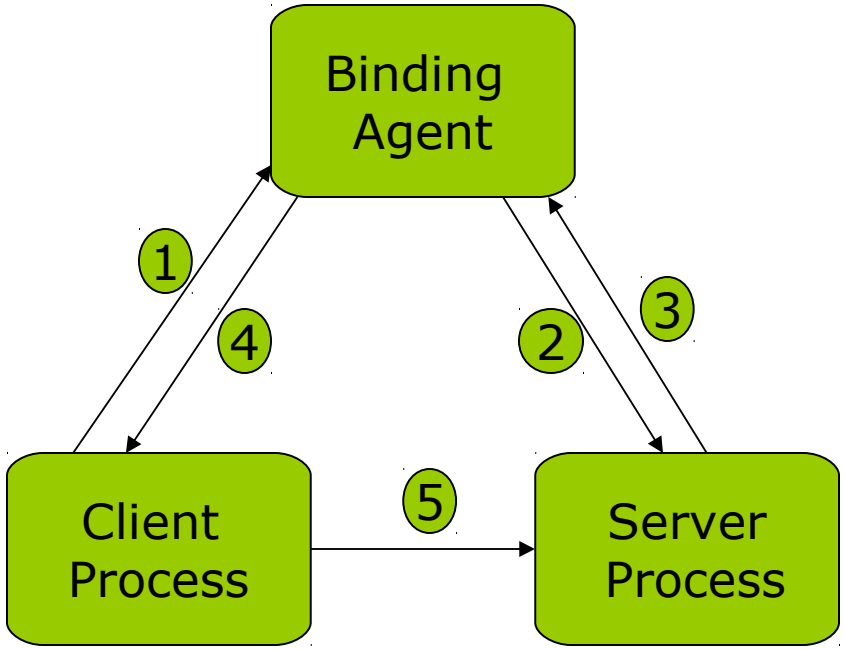
- Binding at compile time
  - Servers network address can be compiled into client code.
- Binding at link time
  - Client makes an import request to the binding agent for the service before making a call.
  - Servers handle cached by the client to avoid contacting binding agent for subsequent calls.
  - This method is suitable for those situations in which a client calls a server several times once it is bound to it.

Cont…

- Binding at call time
  - A client is bound to a server at the time when it calls the server for the first time during its execution.
  - The commonly used approach for binding at call time is the indirect call method.

Binding at call time by the method of indirect call :

1. The client process passes the server's interface name and the arguments of the RPC calls to the binding agent.
2. The binding agent sends an RPC call message to the server, including in it the arguments received.
3. The server returns the result of request processing to BA.
4. The BA returns this result to the client along with the server's handle.
5. Subsequent calls are sent directly from the client process to the server process.



Changing bindings

- The client or server of a connection may wish to change the binding at some instance of time due to some change in the system state.

- When a binding is altered by the concerned server, it is important to ensure that any state data held by the server is no longer needed or can be duplicated in the replacement server.

Security issues

- Is the authentication of the server by the client required?
- Is the authentication of the client by the server required when the result is returned?
- Is it all right if the arguments and results of the RPC are accessible to users other than the caller and the callee?

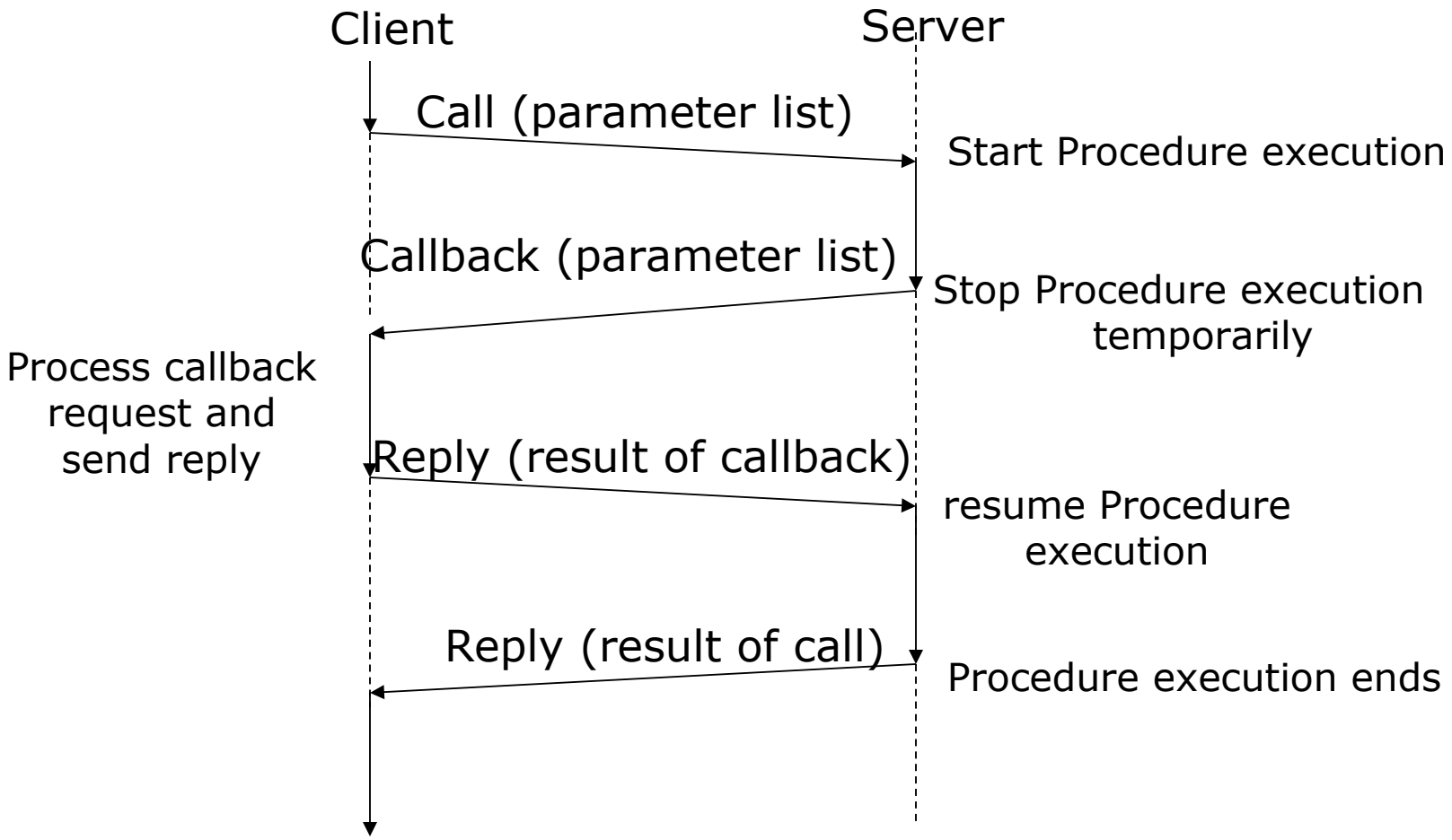
Some special types or RPCs

- Callback RPC
- Broadcast RPC
- Batch-mode RPC

Callback RPC

- It facilitates a peer-to-Peer paradigm among participating processes.
- It allows a process to be both a client and a server.
- Remotely processed interactive Application
- Issues
  - Providing server with clients handle
  - Making the client process wait for the callback RPC
  - Handling callback deadlocks

Cont…


Client
Server
Call (parameter list)
Start Procedure execution
Callback (parameter list)
Stop Procedure execution temporarily
Process callback request and send reply
Reply (result of callback)
resume Procedure execution
Reply (result of call)
Procedure execution ends

Broadcast RPC

- A client's request is broadcast on the network and is processed by all servers that have the procedure for processing that request.

- Methods for broadcasting a client's request:
  1. Use of broadcast primitive to indicate that the client's request message has to be broadcasted.
  2. Declare broadcast ports.

- Back-off algorithm can be used to increase the time between retransmissions.
  - It helps in reducing the load on the physical network and computers involved.

Batch-mode RPC

- Batch-mode RPC is used to queue separate RPC requests in a transmission buffer on the client side and then send them over the network in one batch to the server.
  - It reduces the overhead involved in sending requests and waiting for a response for each request.
  - It is efficient for applications requiring lower call rates and client doesn’t need a reply.
  - It requires reliable transmission protocol.

RPC in heterogeneous environments

- Three common types of heterogeneity that must be considered:
  1. Data representation
  2. Transport protocol
  3. Control protocol

Lightweight RPC (LRPC)

- The communication traffic in OS are of two type
  1. Cross-domain - involves communication between domains on the same machine.
  2. Cross-machine - involves communication between domains located on separate machines.
- The LRPC is a facility designed and optimized for cross-domain communications.
- For cross domain, user level server processes have its own address space.
- LRPC is safe and transparent and represents a viable communication alternative for microkernel operating systems.

Cont…

- Techniques used by LRPC for better performance:
  - Simple control transfer
  - Simple data transfer
  - Simple stubs
  - Design for concurrency

Simple control transfer

- It uses a special threads scheduling mechanism, called handoff scheduling for direct context switch from the client thread to the server thread of an LRPC.

Simple data transfer

- LRPC reduces the cost of data transfer by performing fewer copies of the data during its transfer from one domain to another.
- An LRPC uses a shared-argument stack that is accessible to both the client and the server.
- Pairwise allocation of argument stacks enables LRPC to provide a private channel between the client and server and also allows the copying of parameters and results as many times as are necessary to ensure correct and safe operation.

Simple stub

- Every procedure has a call stub in the client's domain and an entry stub in the server's domain.
- To reduce the cost of interlayer crossings, LRPC stubs blur the boundaries between the protocol layers.

Optimizations for better performance

- Concurrent access to multiple servers
- Serving multiple requests simultaneously
- Reducing per-call workload of servers
- Reply caching of idempotent remote procedures
- Proper selection of timeout values
- Proper design of RPC protocol specification