CHAPTER - 4
REMOTE COMMUNICATION

Topics

- Introduction to Remote Communication
- Remote Procedural Call Basics
- RPC Implementation
- RPC Communication
- Other RPC Issues
- Case Study: Sun RPC
- Remote invocation Basics
- RMI Implementation

Introduction to Remote Communication

Introduction

- Middleware

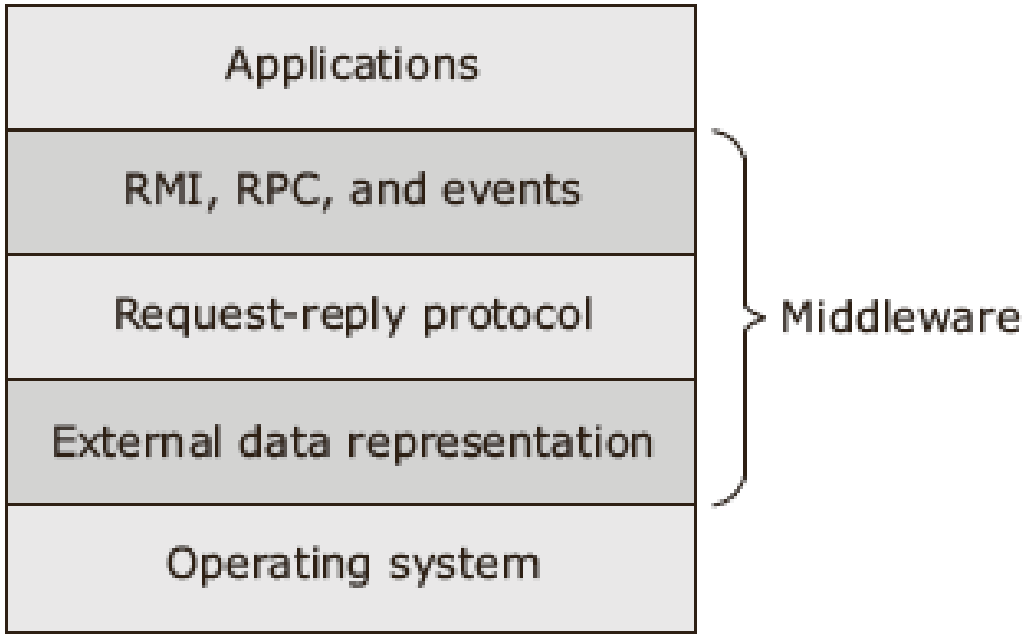


Figure 4-1 Role of middleware in remote communication

Remote Procedural Call Basics

Local Procedure Call

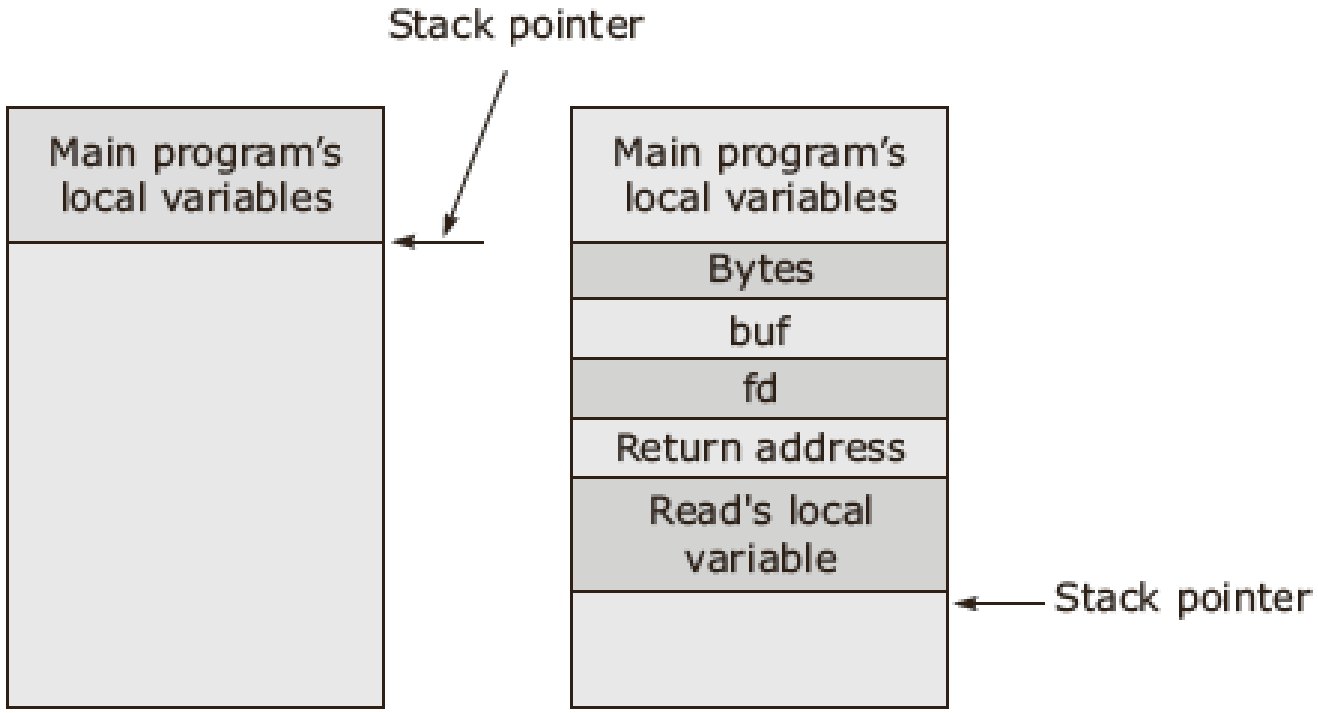


Figure 4-2 Local procedure call

Remote Procedure Call

- Basic RPC operation

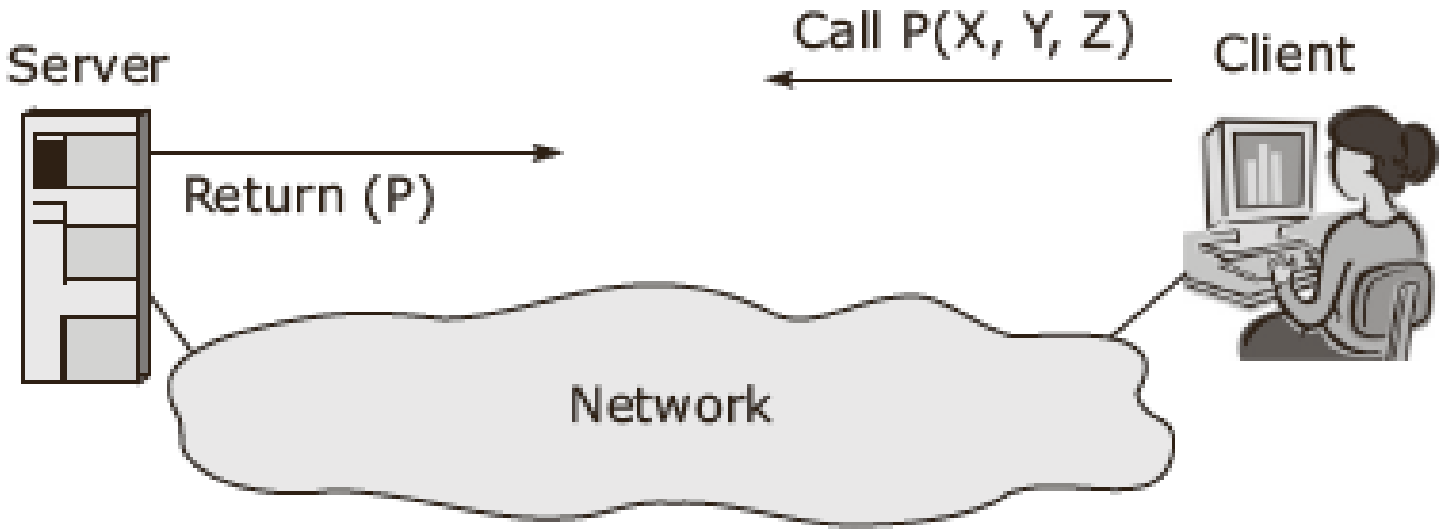


Figure 4-3 Basic RPC model

RPC operation

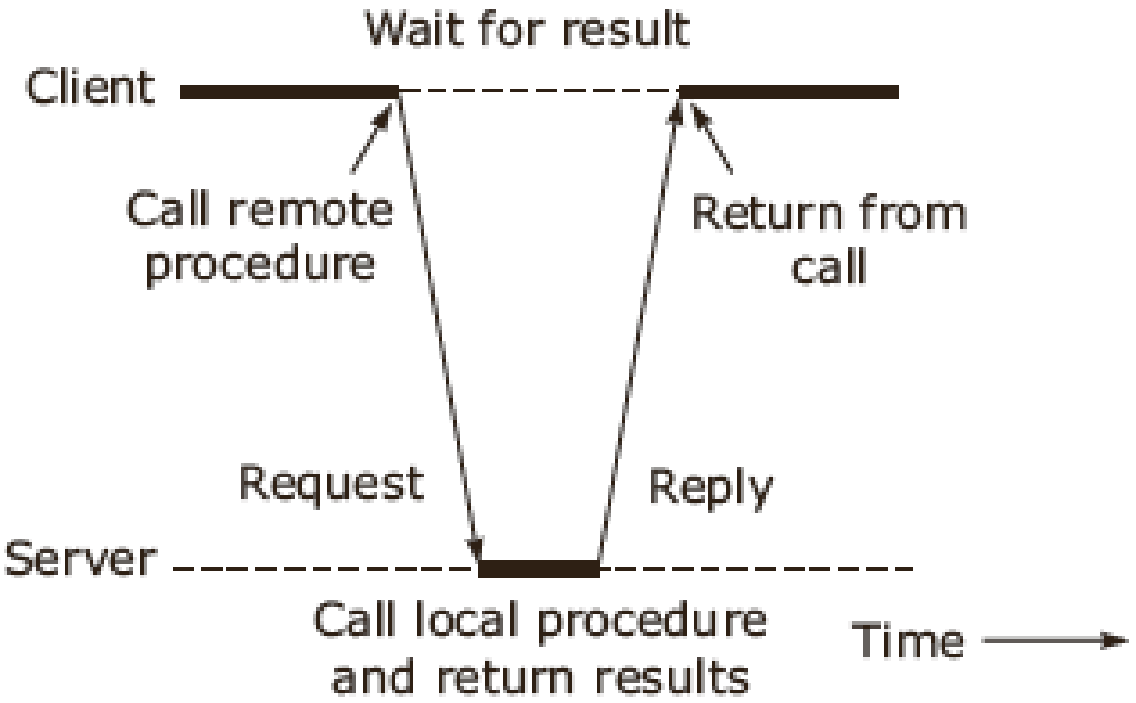


Figure 4-4 A typical RPC

Elements of RPC mechanism implementation

- Client
- Client stub
- RPC Runtime
- Server stub
- Server

RPC Execution

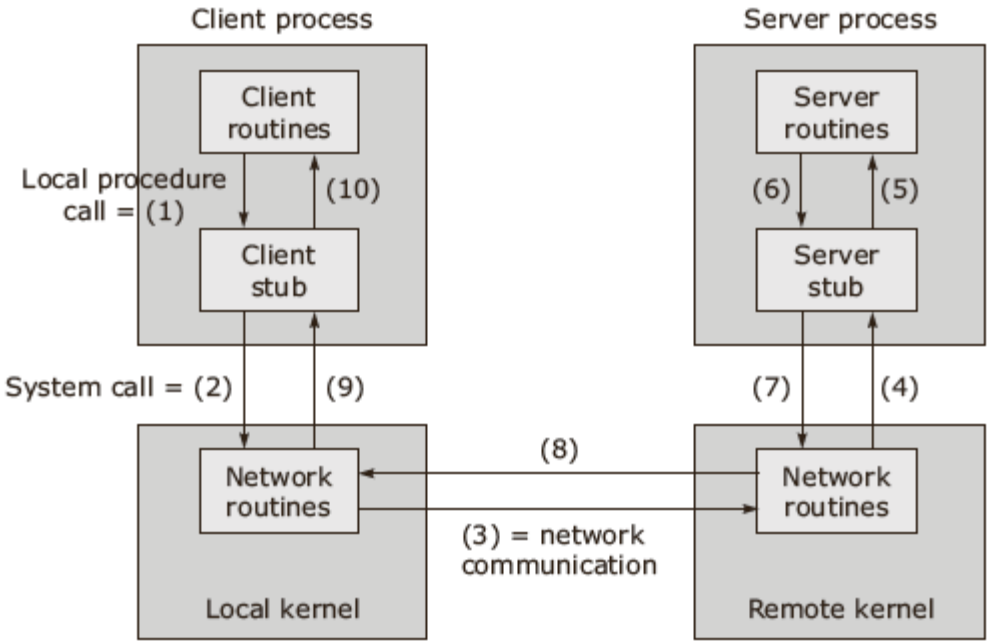


Figure 4-5 RPC execution

Stub generation

- Manual generation
- Auto generation using Interface Definition Language (IDL)

RPC Compilation

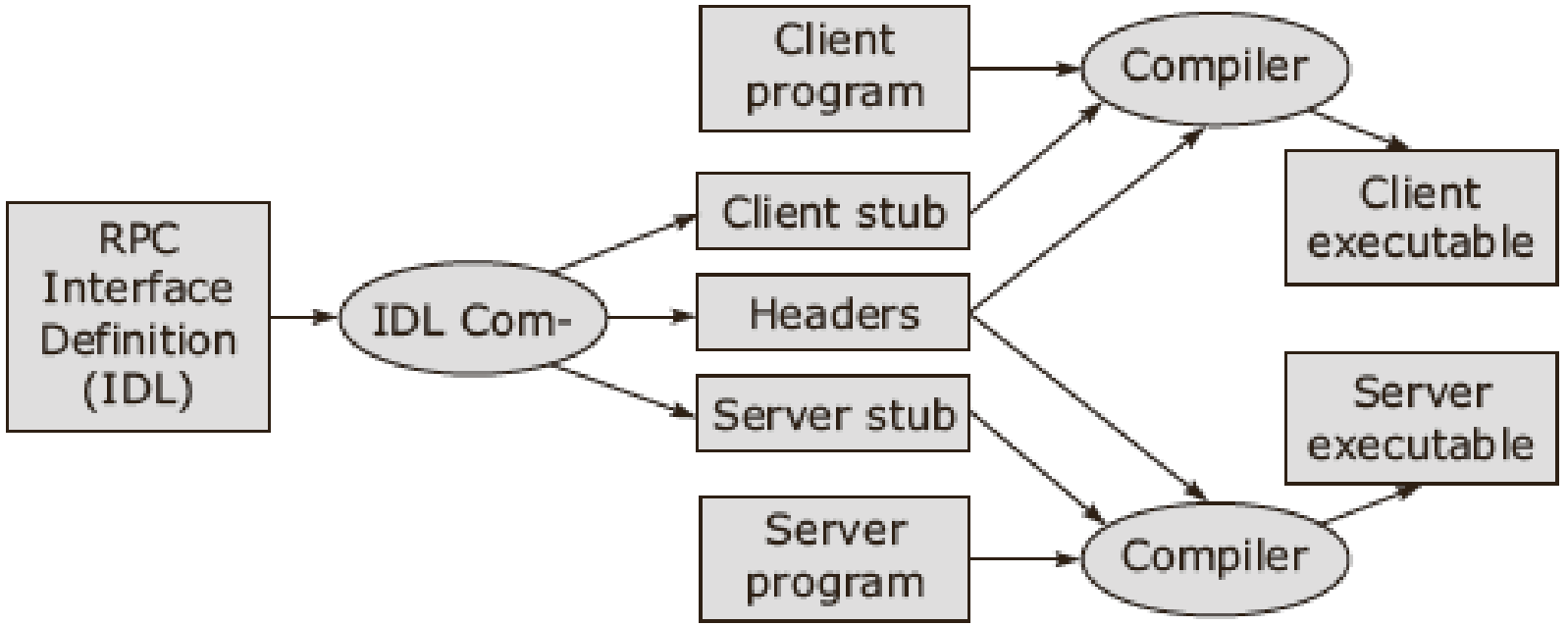


Figure 4-6 Steps for RPC compilation

RPC Implementation

RPC implementation

- RPC messages:
  - Call / Request
  - Reply

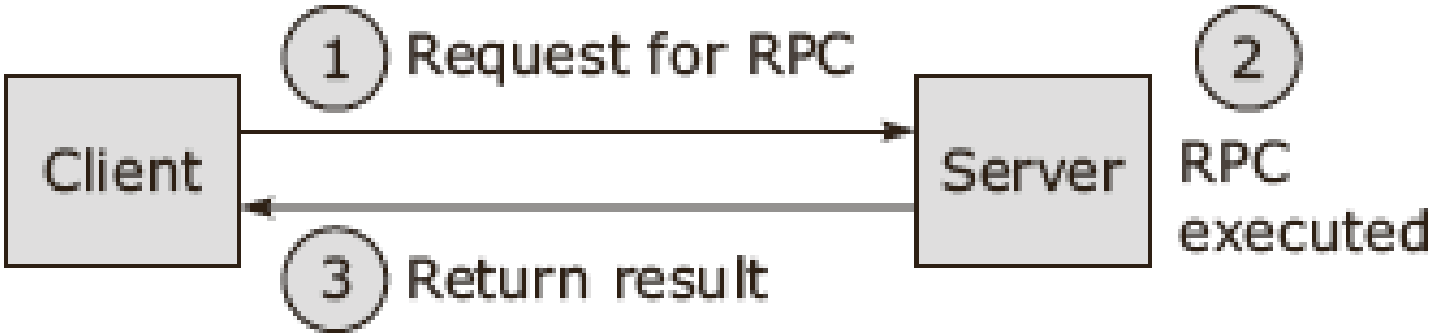


Figure 4-7 RPC messages

RPC Call/ Request message

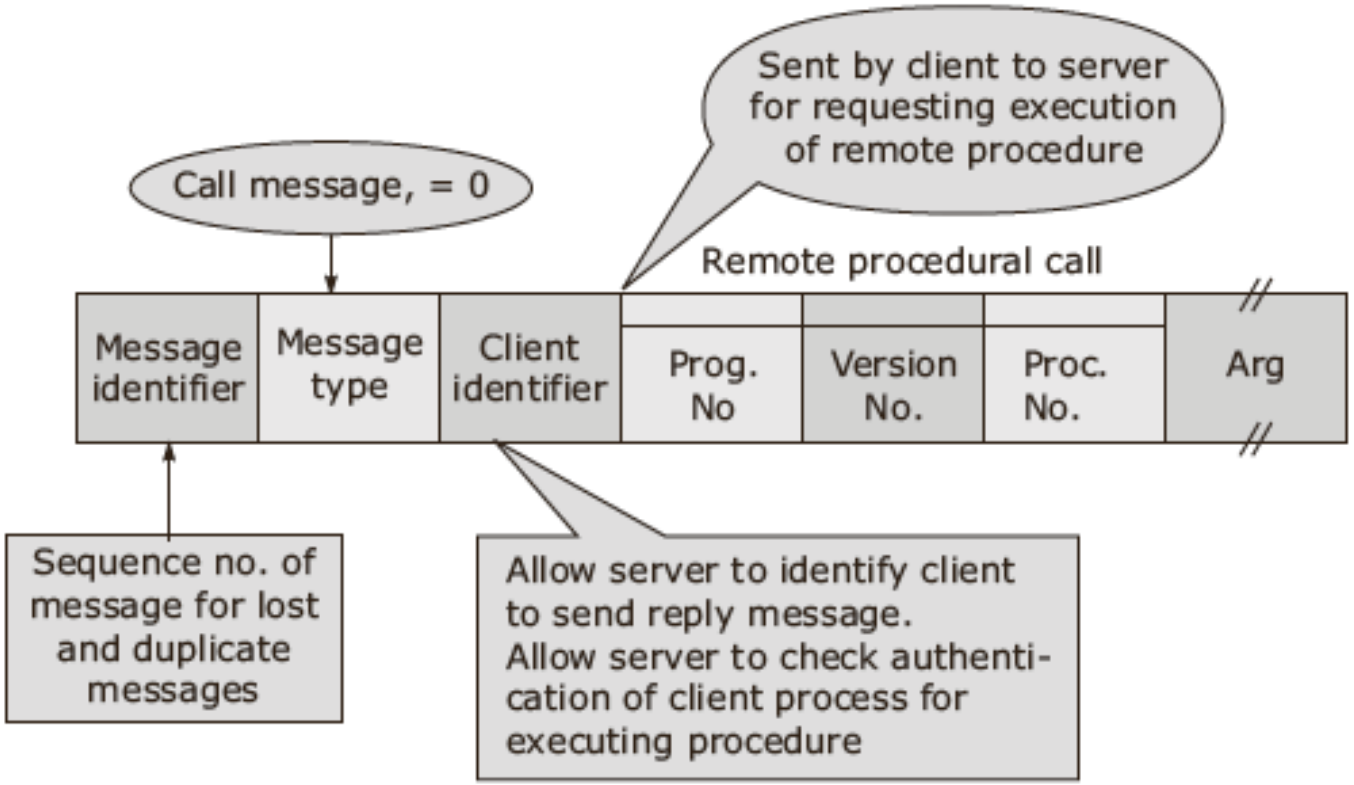


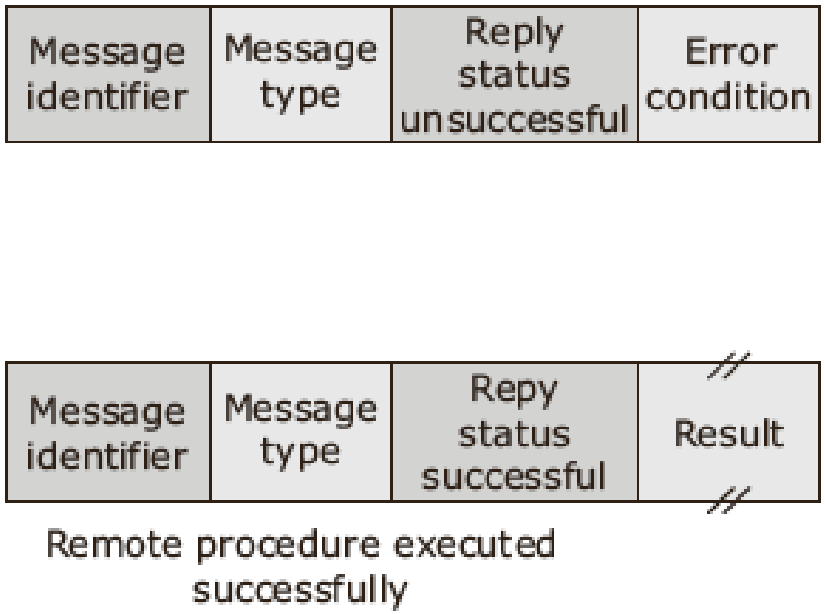
Figure 4-8 RPC call/request message format

RPC reply conditions

Table 4-1 RPC reply message conditions

Condition	Response from the server
Server receives an unintelligible call message, probably because the call message has violated the RPC protocol.	Rejects the call.
Server receives the call messages with unauthorized client ID, i.e. the client is prevented from making the RPC request.	Return reply unsuccessful, and does not execute RPC.
Server does not any receive procedure ID information from the message ID field—program number, version number, or ID.	Return reply unsuccessful, and does not execute RPC.
If all the above conditions are satisfied, the server executes the RPC, but may not be able to decode its arguments due to incompatible RPC interface.	Return reply unsuccessful and does not execute RPC.
Server executes the RPC but an exception condition occurs.	Return reply unsuccessful.
Server executes the RPC successfully without any problems.	RPC is successful and the server returns the result.

RPC reply message



Error conditions

1. Call message not intelligible (RPC protocol violated)
2. Unauthorized to use service
3. Server finds the remote program, version, procedure number are not available with it.
4. Unable to decode supplied arguments
5. During execution, an exception condition occurs

Figure 4-9 RPC reply message format

Parameter Passing Semantics

- Call-by-value semantic
  - Marshalling
- Call-by –reference semantic
- Call-by-copy/restore semantic

Call-by-value copies all parameters into a message before transmission . Call-by-reference passes pointers to the parameters that are passed from the client to the server. Call-by-copy/restore uses temporary storage accessible to both programs

Call-by-value semantic

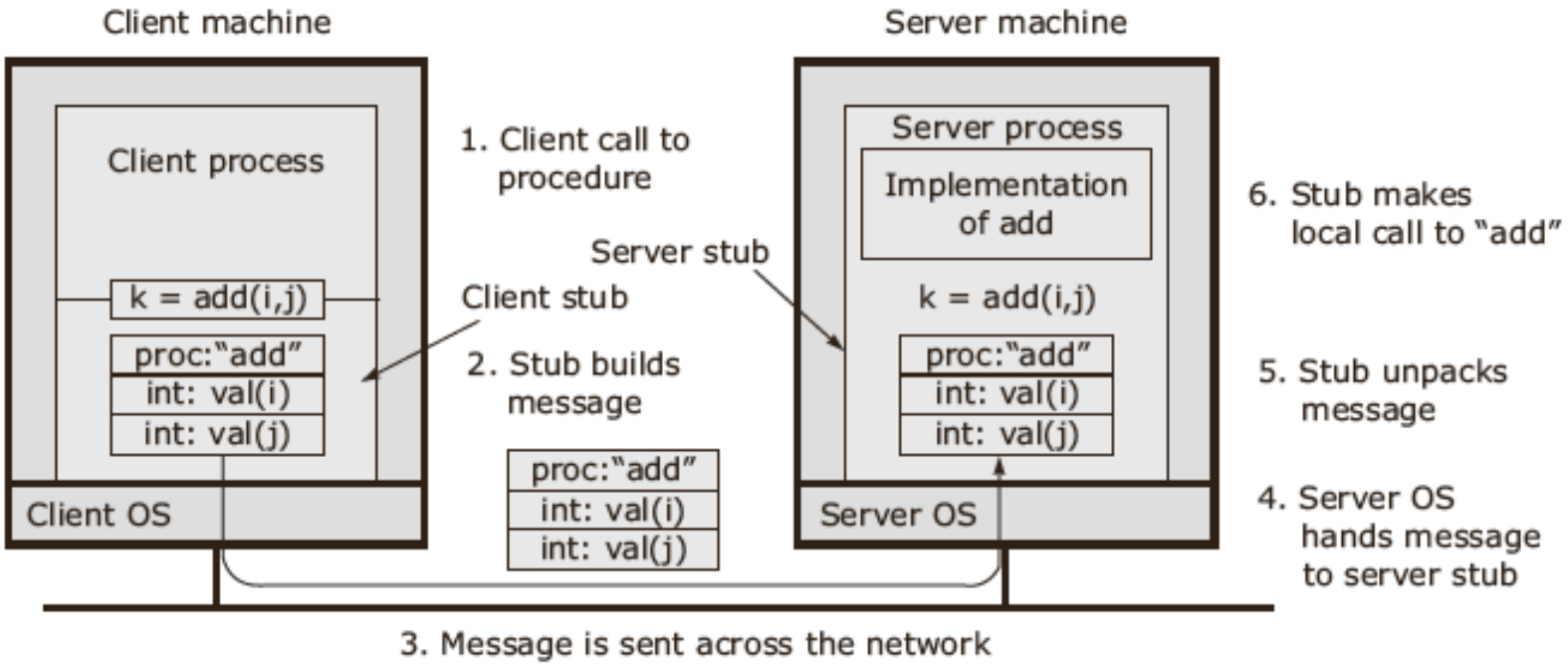


Figure 4-10 Example of call-by-value semantic

Byte ordering

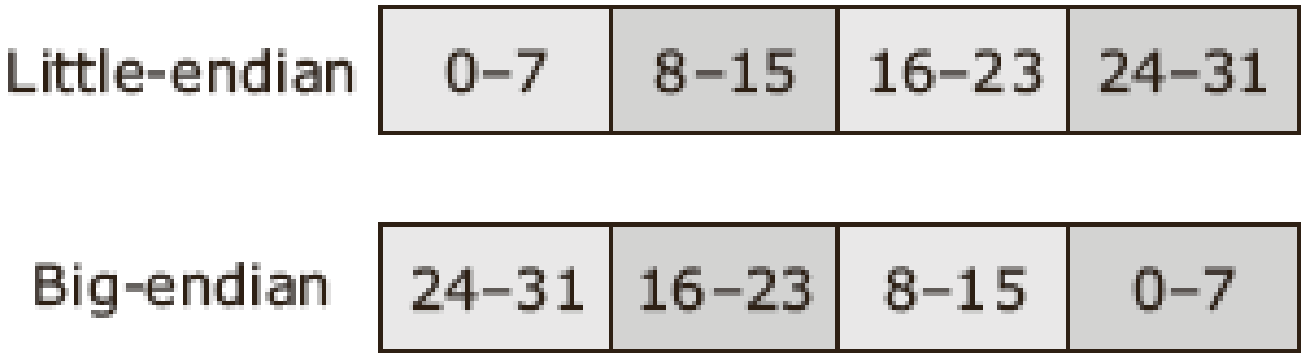


Figure 4-11 Data representations

Server management

- Server implementation
  - Stateless server
  - Stateful server
- Server management
  - Instance per call
  - Instance per session
  - Persistent servers

RPC communication

- RPC call semantics

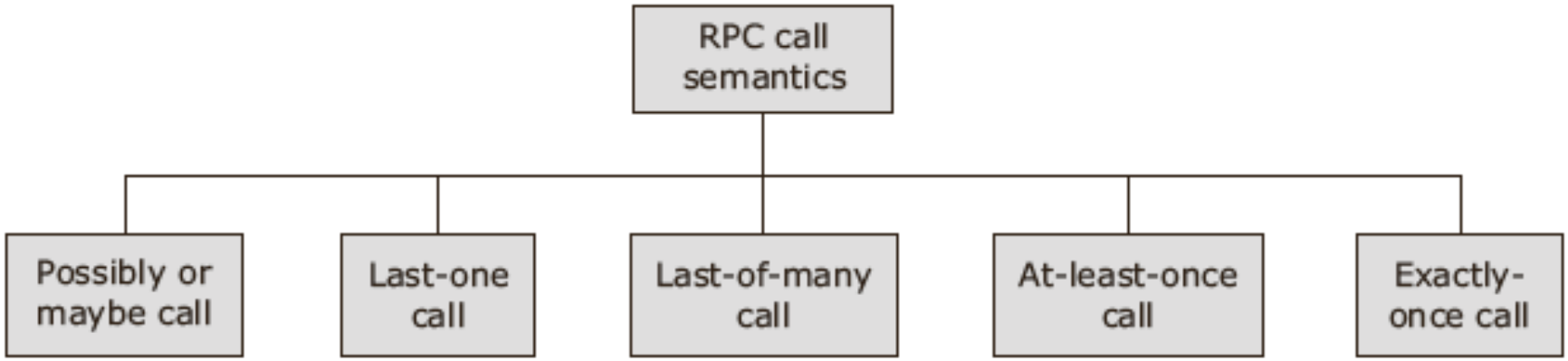


Figure 4-12 RPC call semantics

Orphan calls

- Calls whose caller has expired due to a node crash
- Handle orphan calls by using:
  - Extermination
  - Reincarnation
  - Gentle reincarnation
  - Expiration

RPC communication protocols

- Request protocol
- Request/Reply protocol
- Request/Reply/ Acknowledge- Reply protocol

Request protocol

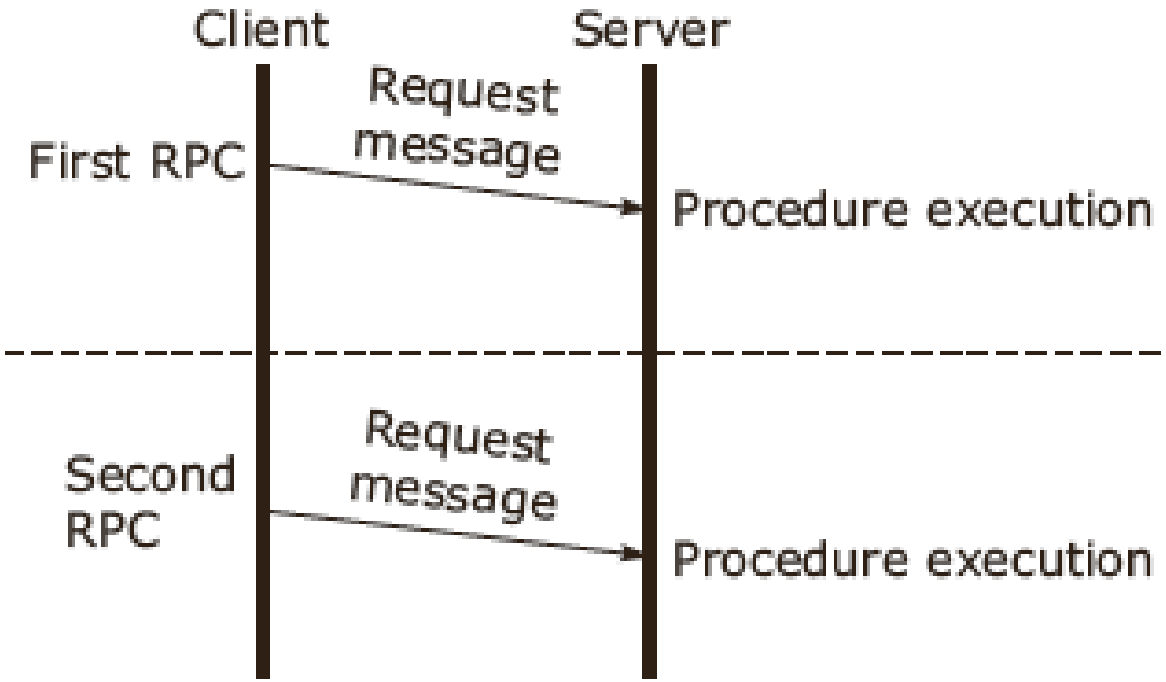


Figure 4-13 The request protocol

Asynchronous RPC

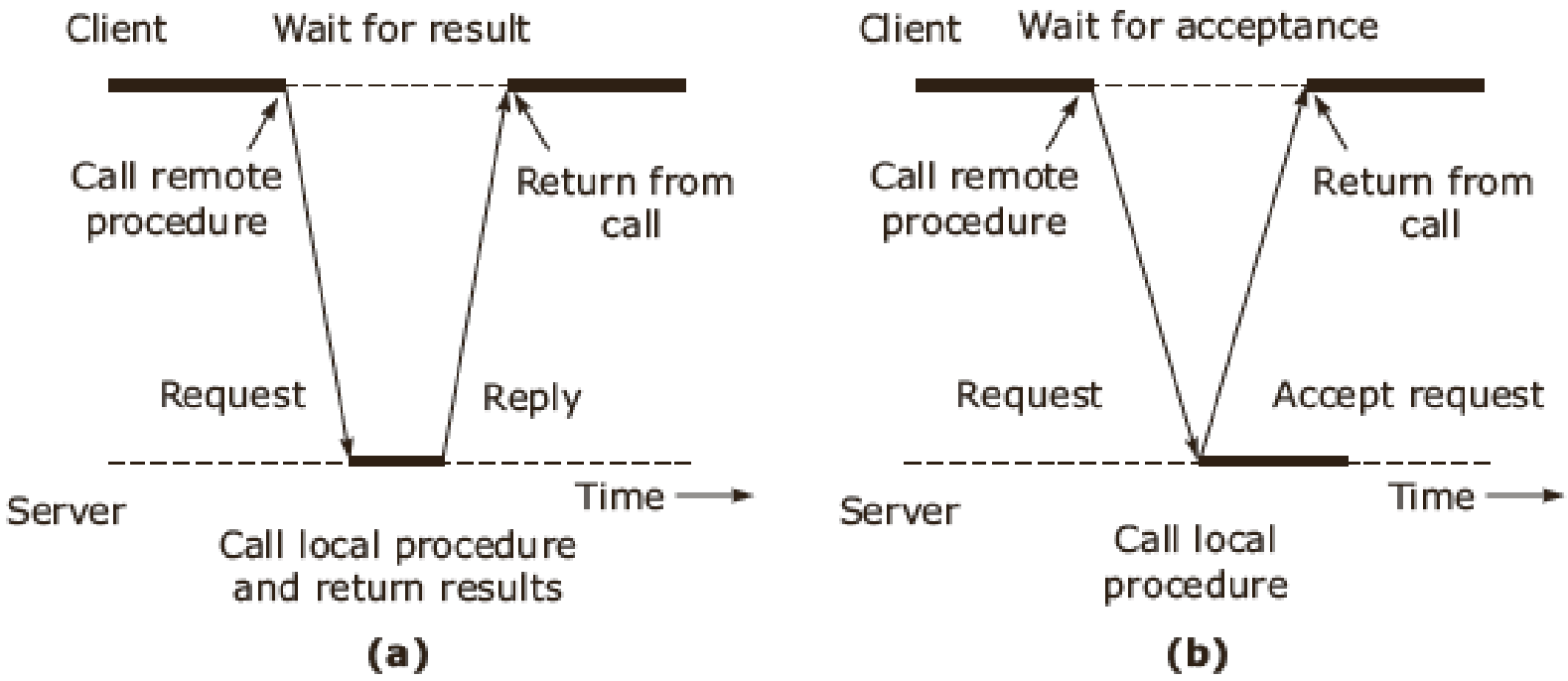


Figure 4-14 Asynchronous RPC

Request/Reply protocol

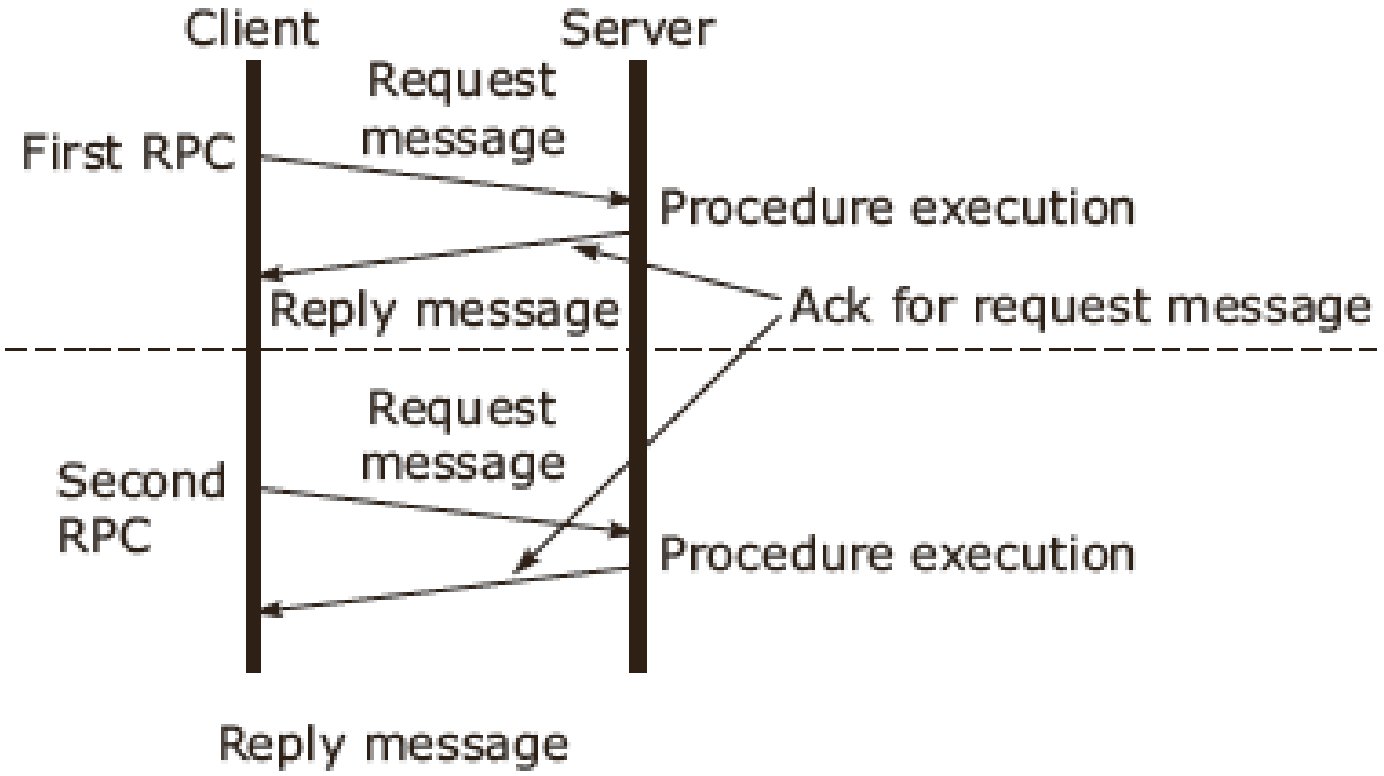


Figure 4-15 The request/reply protocol

Request/Reply/ Acknowledge- Reply protocol

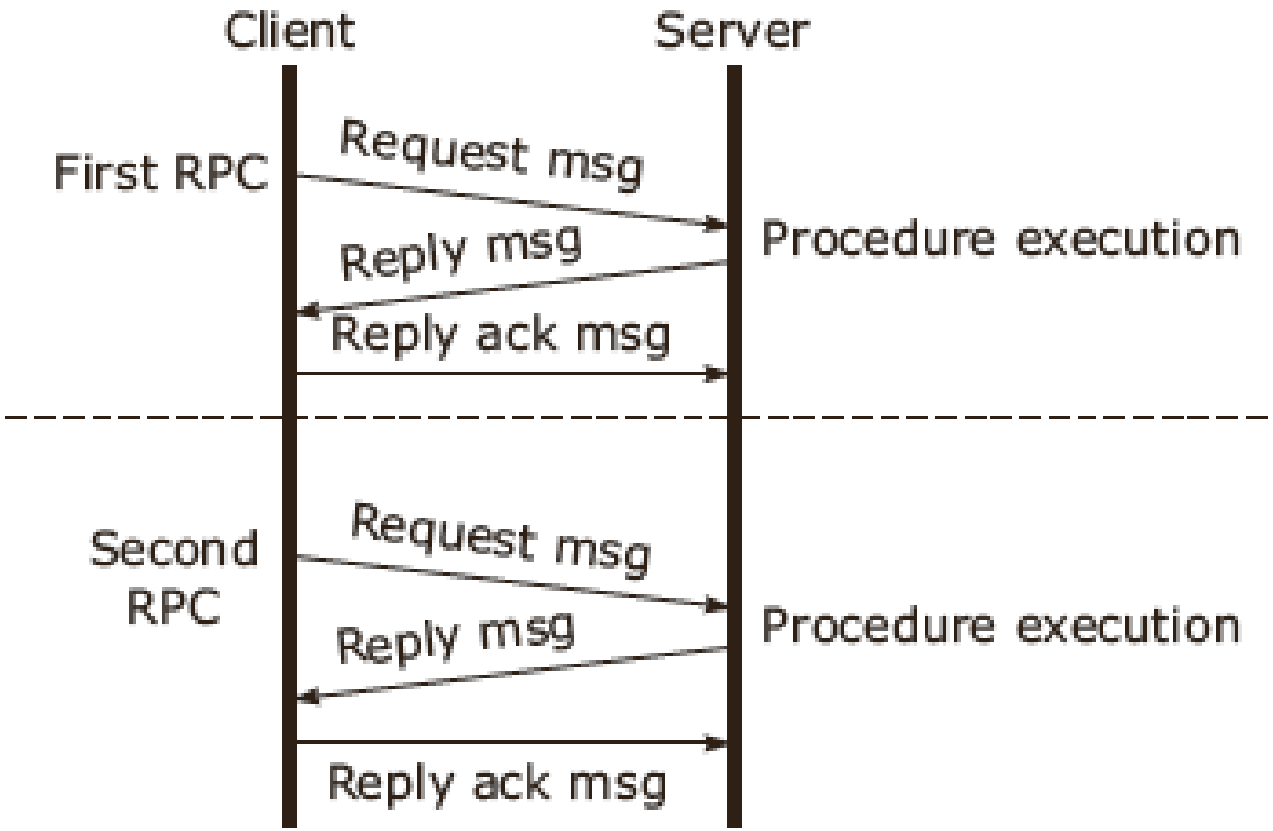


Figure 4-16 The RRA protocol

Client server binding process

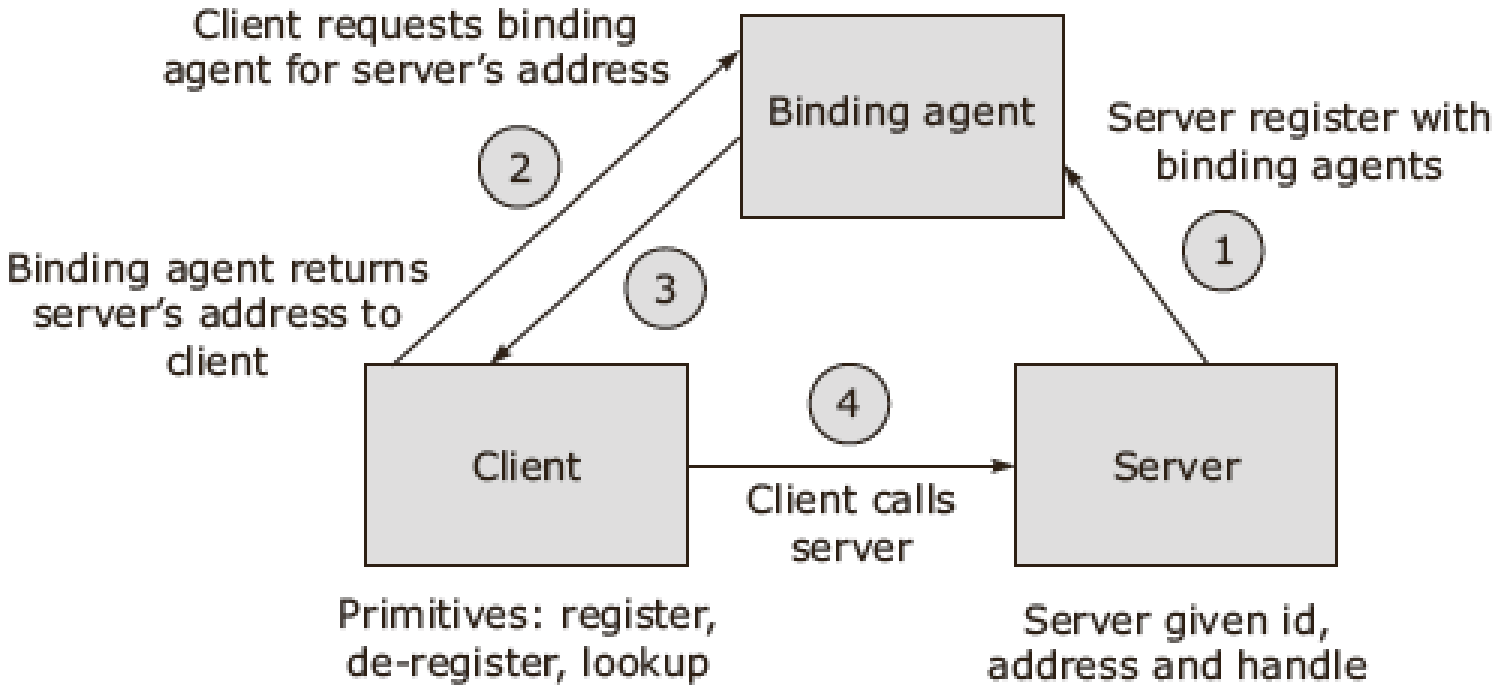


Figure 4-17 Client-server binding

Client Server binding

- Issues
  - Server naming
  - Server locating
- Binding agent primitives
  - Register
  - Deregister
  - Lookup
- Types of binding
  - Fixed binding
  - Dynamic binding
    - At compile time
    - At link time
    - At run time

Other RPC Issues

Other issues in RPC implementation

- Exception handing and security
- Failure handling
- Optimizing RPC execution
- Various types of complicated RPCs

RPC in heterogeneous environment

- Data presentation
- Transport protocol
- Control protocol

Failure handling mechanism in RPC

- Client cannot find the server
- Request from client to the server is lost
- Reply from server to the client is lost
- Server crashes after getting the request
- Client crashes after sending the request

RPC Optimization

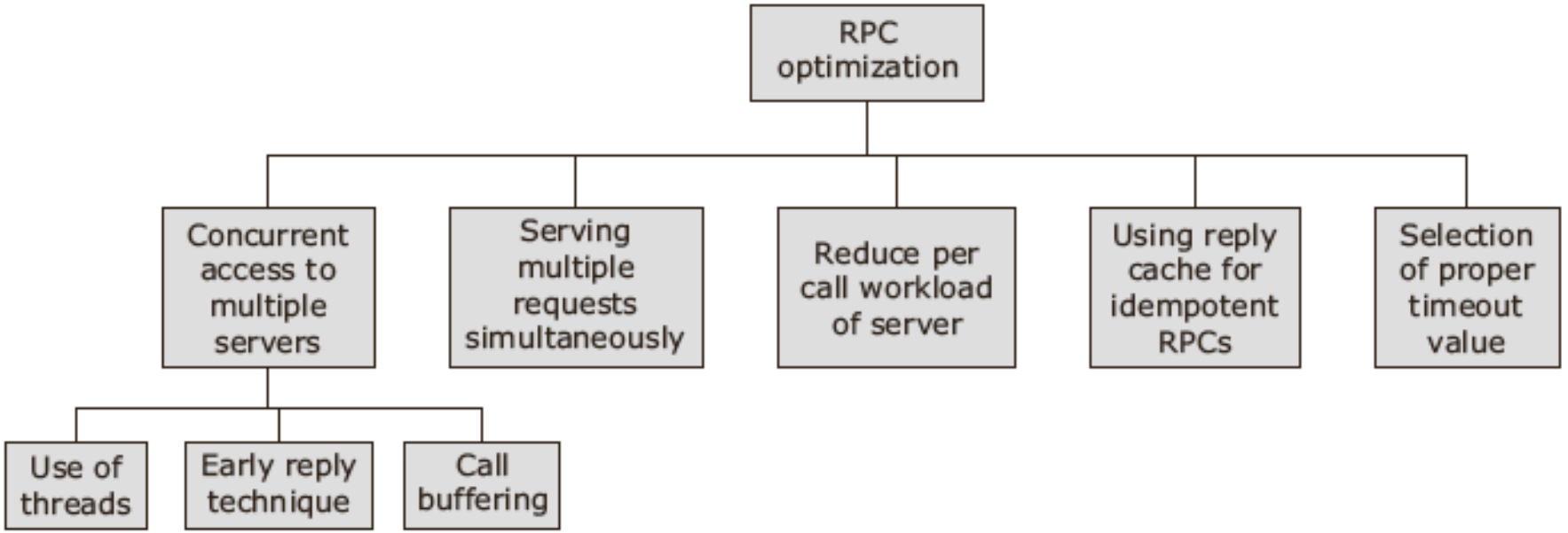


Figure 4-18 Techniques for RPC optimization

Concurrent access to multiple servers

- Use of threads
- Early reply technique
- Call buffering approach
- Serving multiple requests simultaneously
- Reducing call workload of server
- Using reply cache for idempotent RPC

Early Reply technique

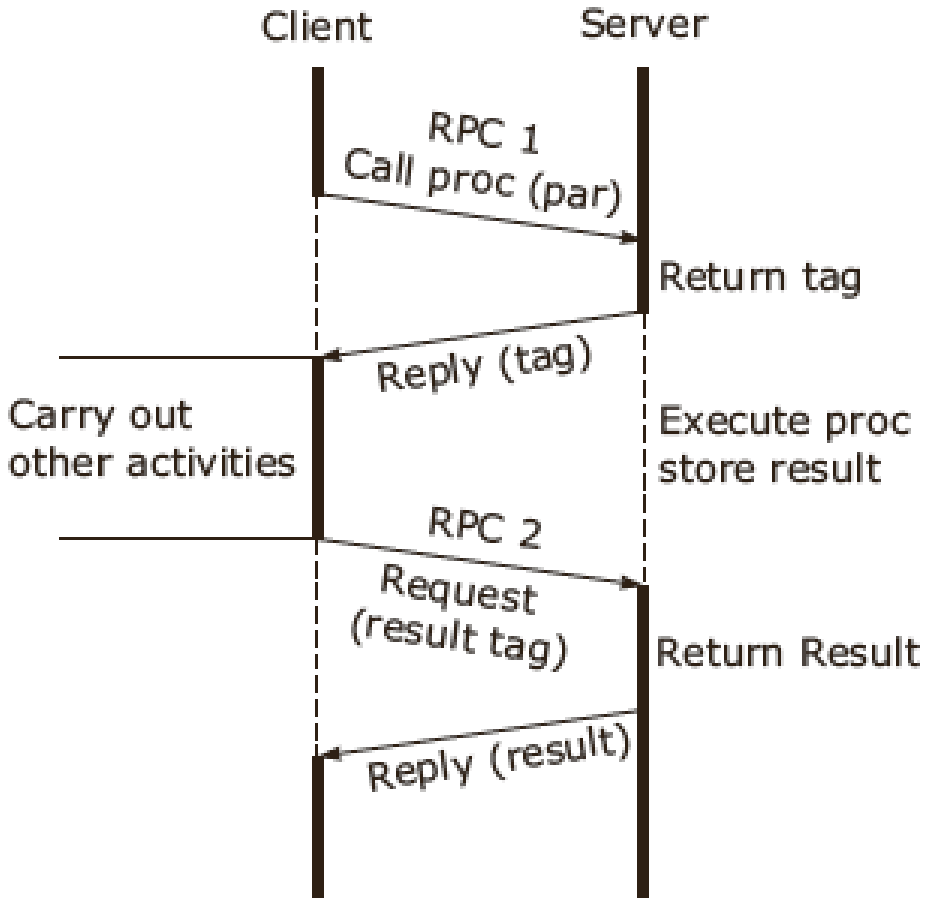


Figure 4-19 Early reply technique

Call buffer approach

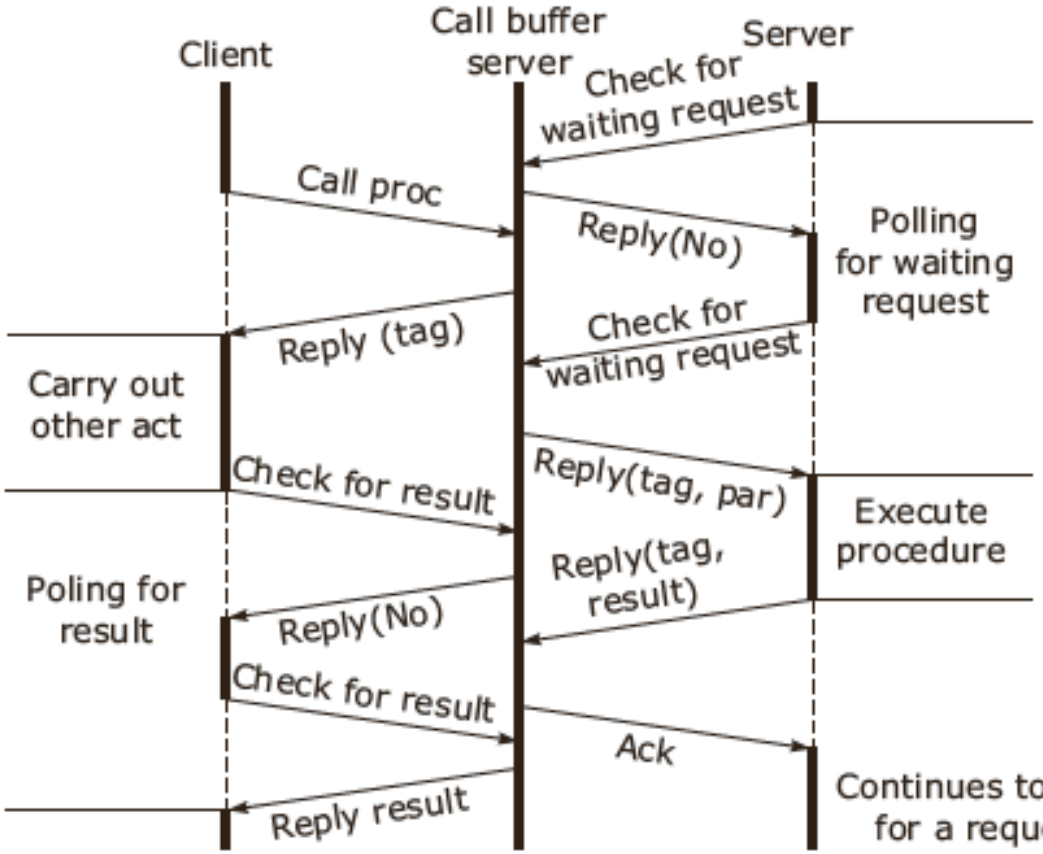


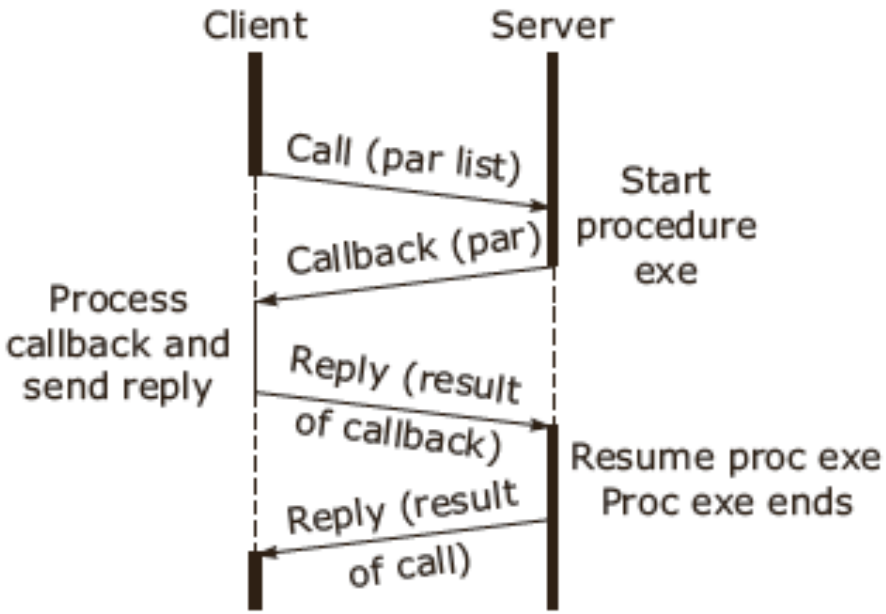
Figure 4-20 Call-buffer approach for concurrent access to multiple servers

Complicated and special RPCs

- Complicated RPCs
  - RPCs with long duration calls or with gaps between calls
  - RPCs with long messages
- Special RPCs:
  - Call back RPC
  - Broadcast RPC
  - Batch mode RPC

Call back RPC

- Client handle is provided to the server
- Client process should wait for callback RPC
- Handle callback deadlocks



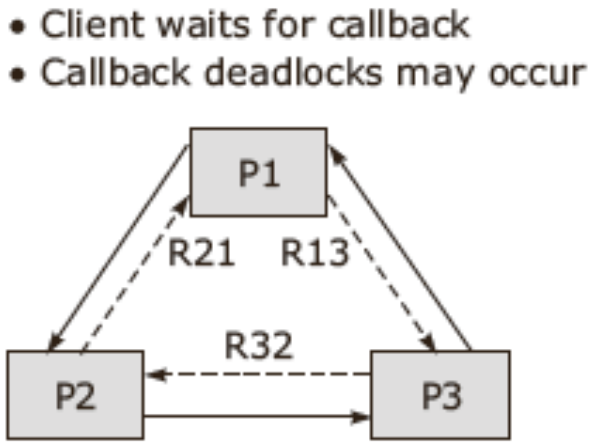


Figure 4-21 Callback RPC

Case Study: Sun RPC

Case Study- Sun RPC

- Uses rpcgen compiler which generates
  - Header file
  - XDR filter file
  - Client stub file
  - Server stub file

Remote invocation Basics

Remote Object Invocation

- Distributed object concept
  - Remote objects reference
  - Remote interface

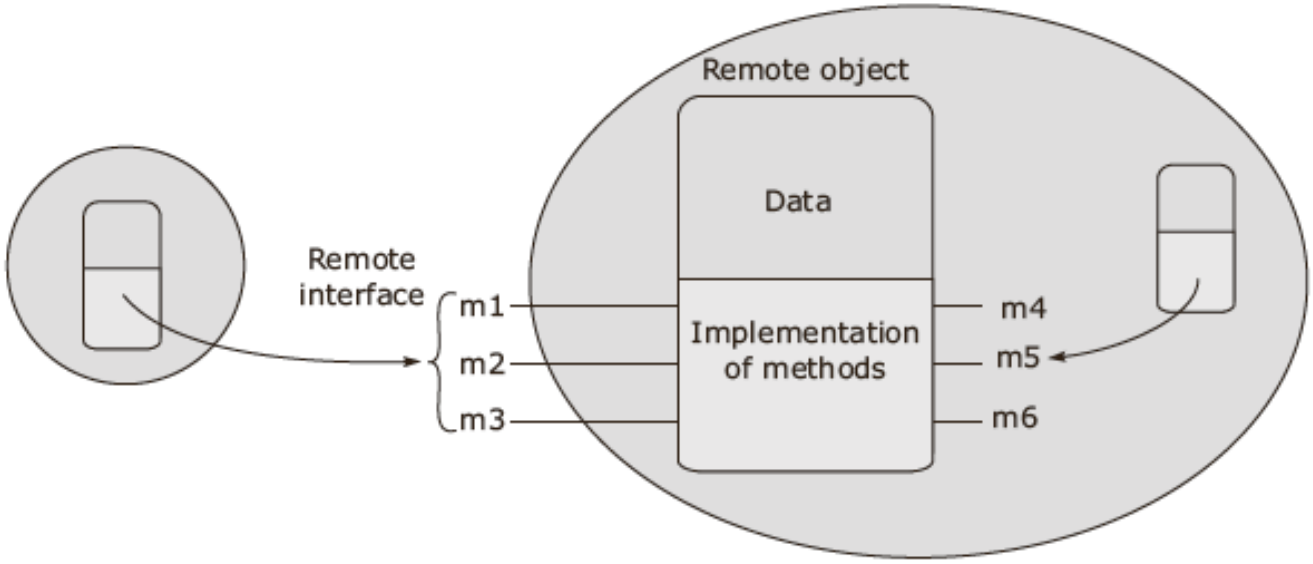


Figure 4-22 Remote object and remote interface

RMI

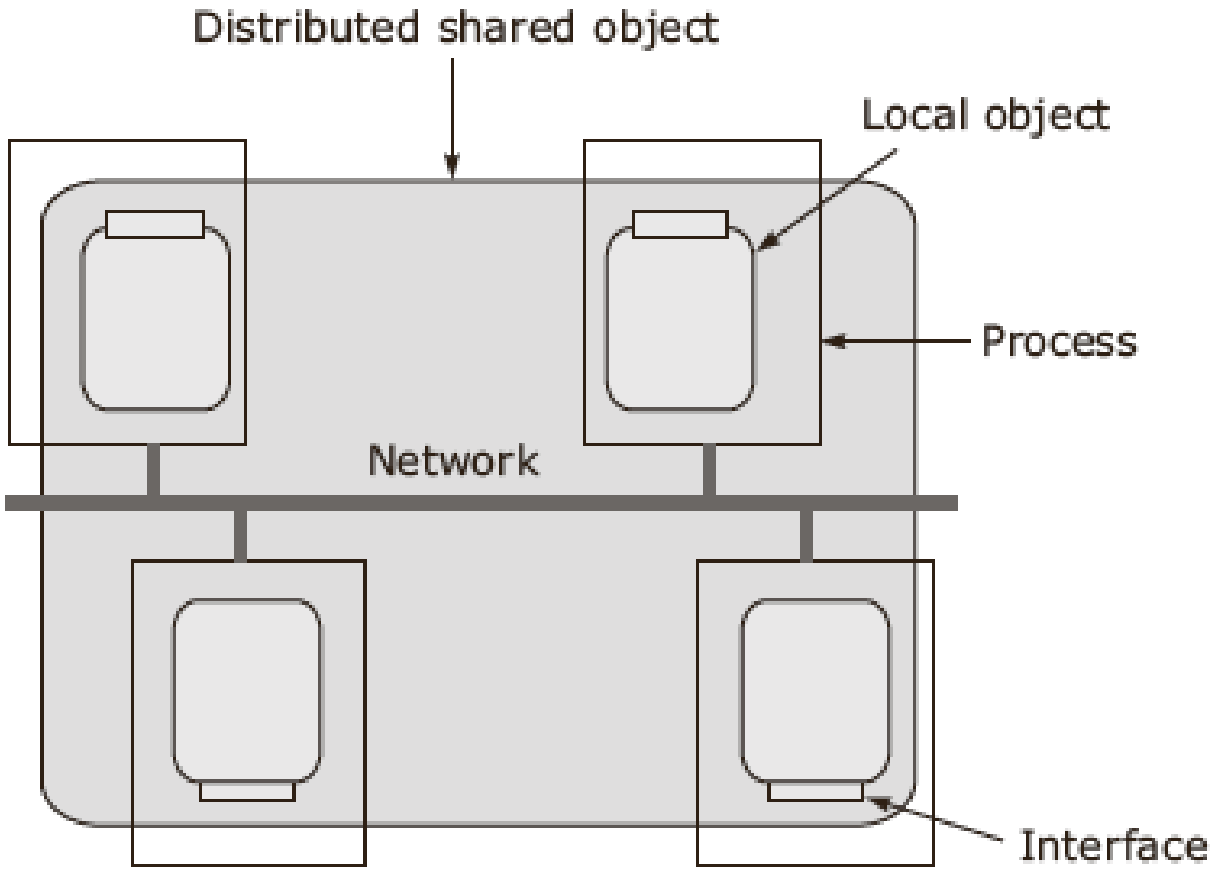


Figure 4-23 Distributed shared object

RMI vs LMI

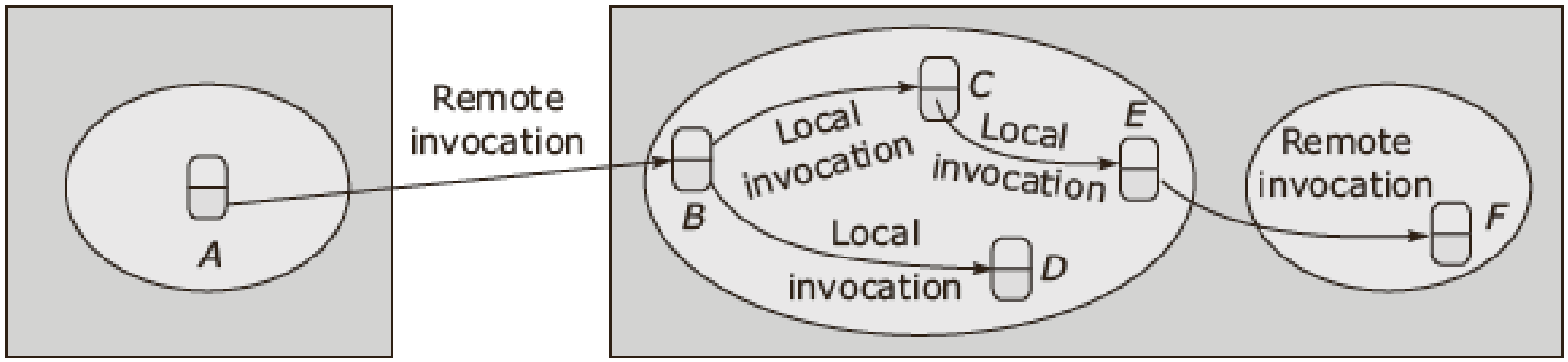


Figure 4-24 RMI and LMI

RMI Implementation

Design issues in RMI

- RMI invocation semantics
- Level of transparency
  - Marshalling
  - Message passing
  - Task of locating and contacting the remote object for the client

- RMI invocation semantics
  - Maybe semantics
  - At-least-once semantics
  - At-most-once semantics

Invocation semantics

Table 4-2 Invocation semantics

Fault tolerance measures			
Retransmit requestmassage	**Duplicate filtering**	**Re-execute procedure of retransmit reply**	**Invocation semantics**
No	Not applicable	Not applicable	Maybe
Yes	No	Re-execute procedure	At-least-once
Yes	Yes	Retransmit reply	At-most-once

Level of Transparency

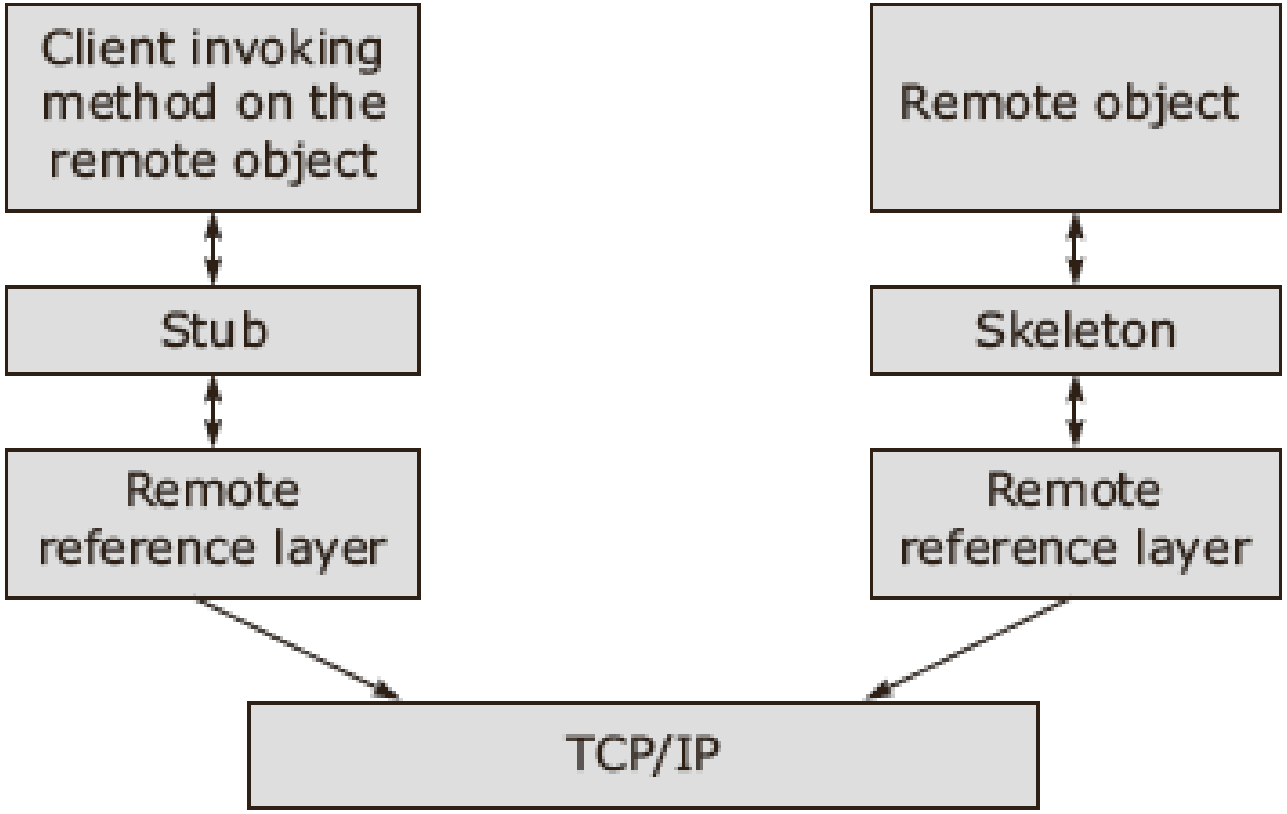


Figure 4-25 RMI flow diagram

Components of RMI

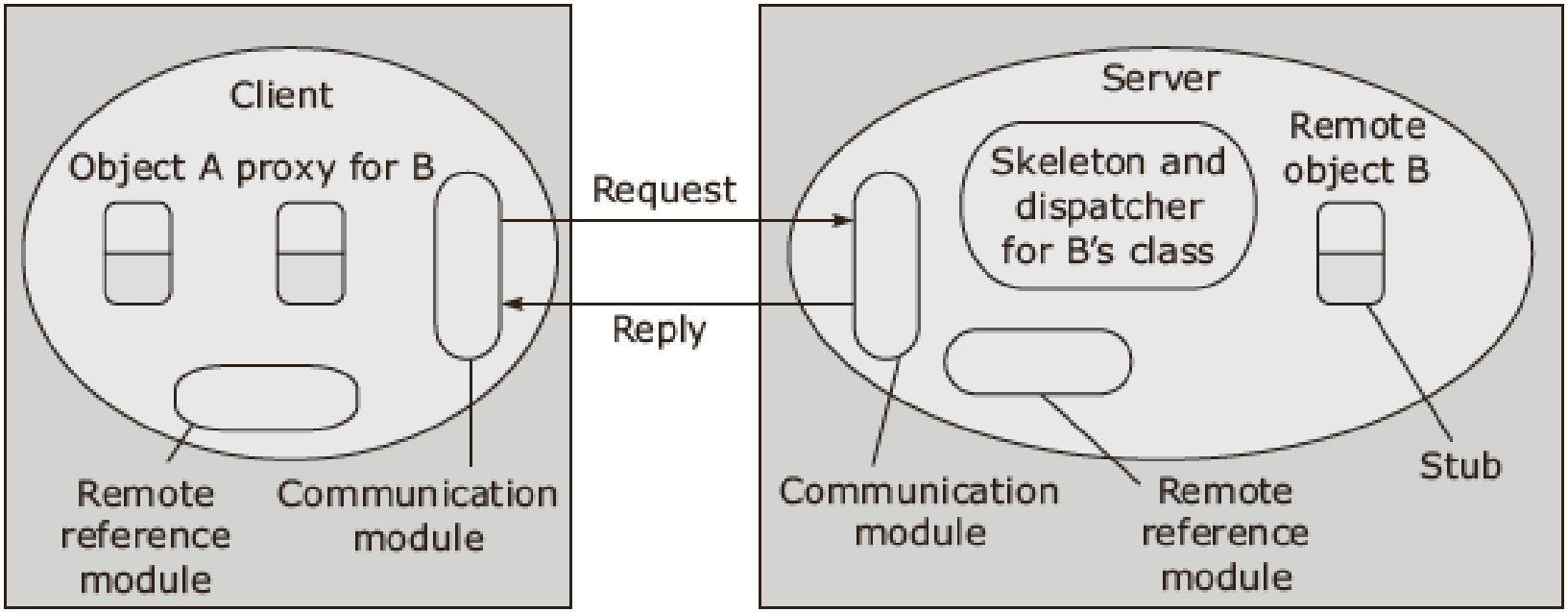


Figure 4-26 RMI components

RMI execution components

- Communication module
- Remote reference module
- RMI software
- Server program
- Client program
- Binder

RMI execution

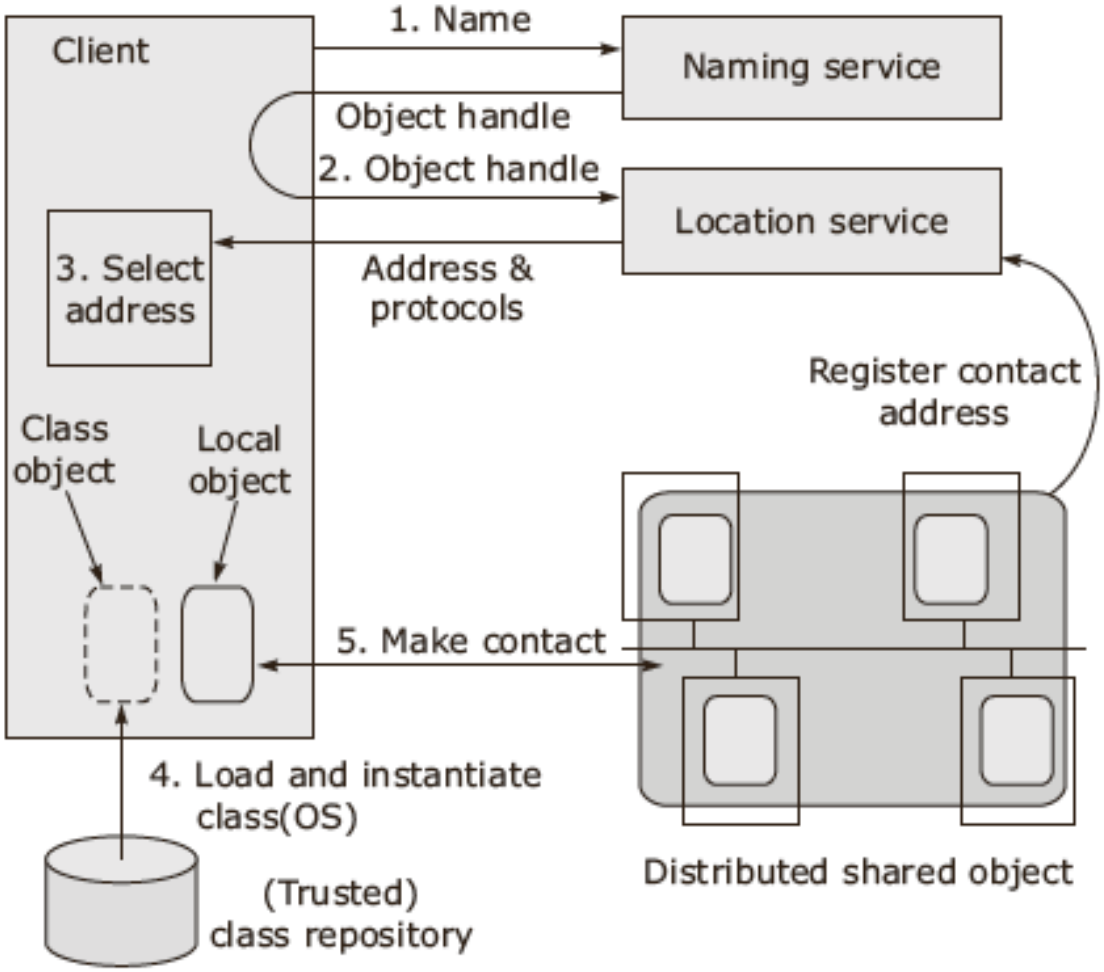


Figure 4-27 RMI implementation

RMI software

- Proxy
- Dispatcher
- Skeleton

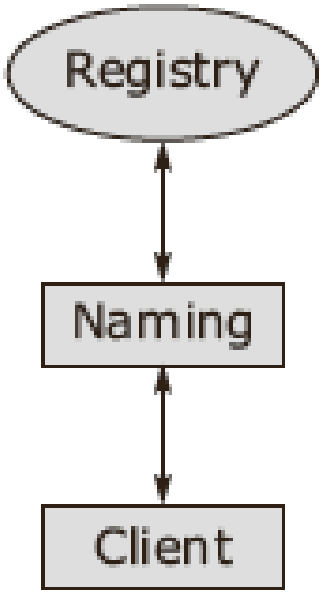


Figure 4-28 Locating remote objects

Types of objects

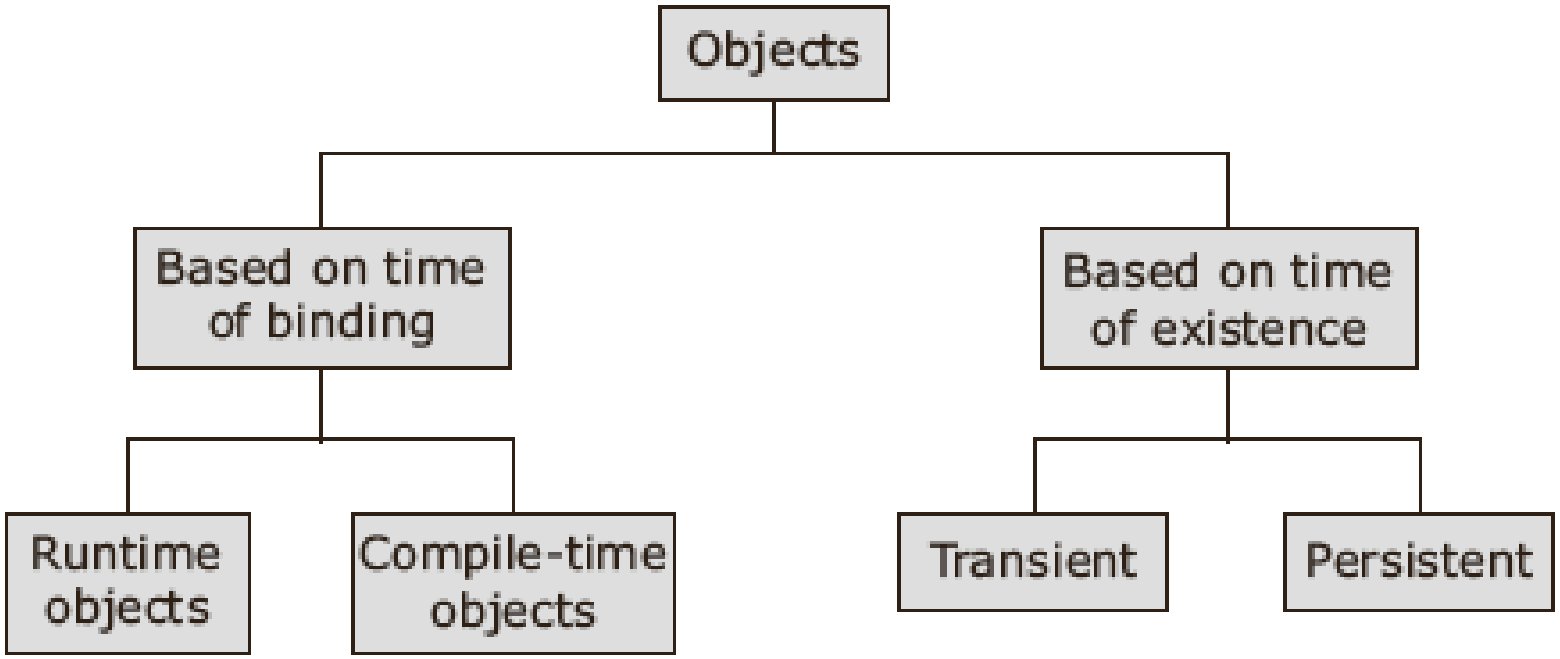


Figure 4-29 Types of objects

Remote invocation readiness

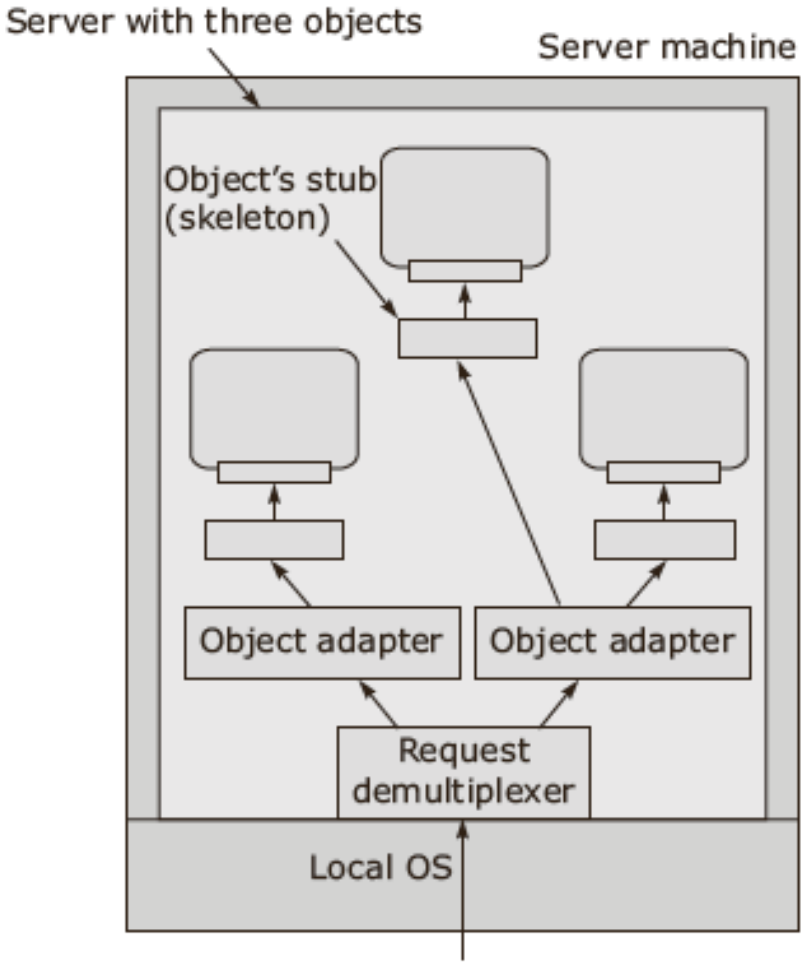


Figure 4-30 Object adapter

RMI binding

- Implicit binding
- Explicit binding

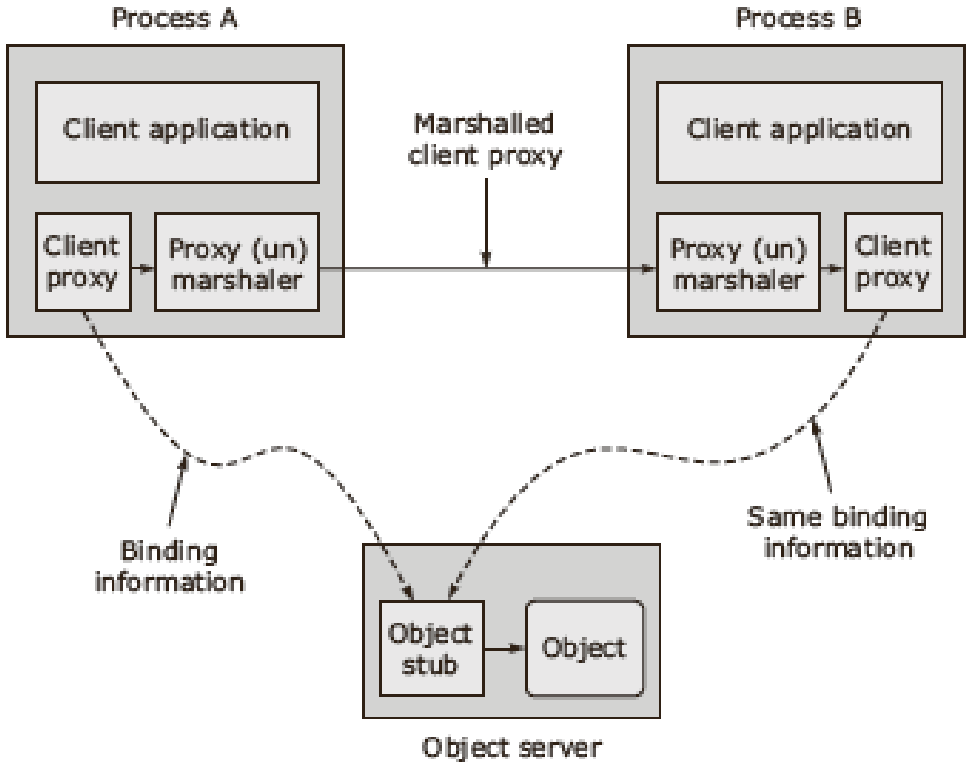


Figure 4-31 Binding

Parameter passing in RMI

- Pass by value
- Pass by reference

Case study: Java RMI

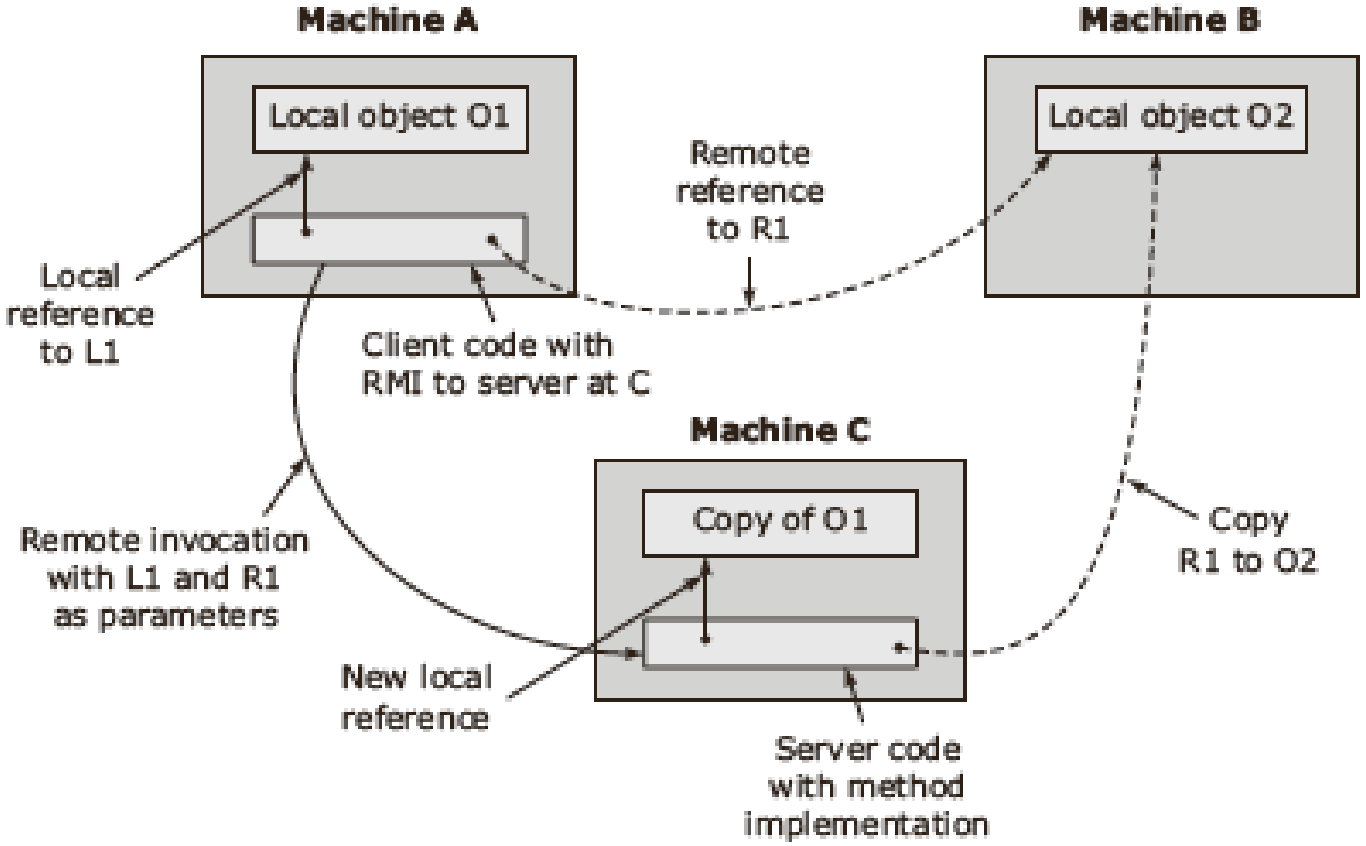


Figure 4-32 Java remote object

Java RMI layer

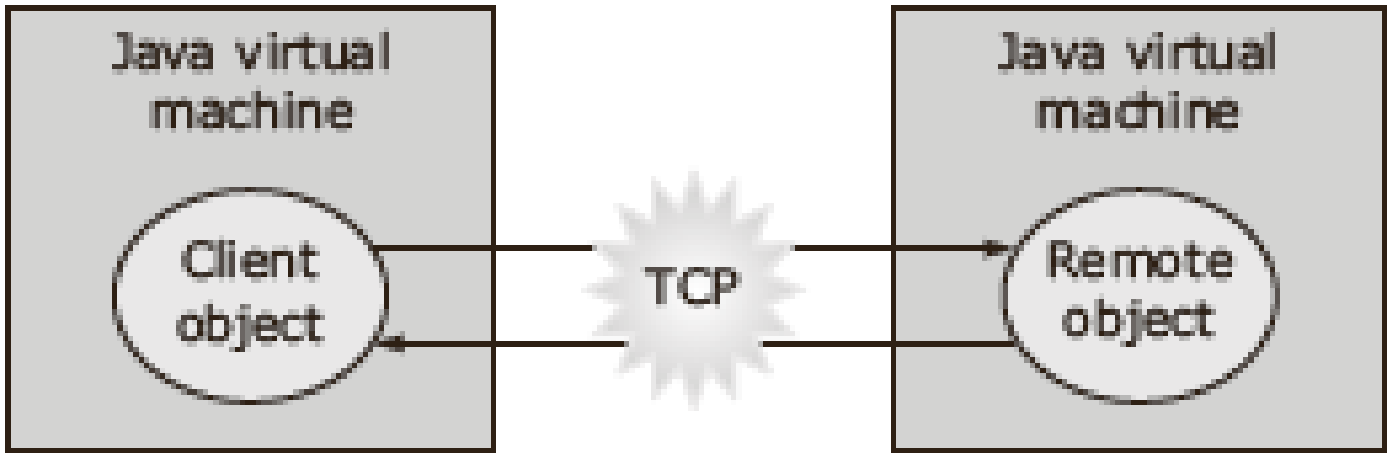


Figure 4-33 Java RMI layer

Summary

- Introduction to Remote Communication
- Remote Procedural Call Basics
- RPC Implementation
- RPC Communication
- Other RPC Issues
- Case Study: Sun RPC
- Remote invocation Basics
- RMI Implementation