Lecture 05 · SENG 41283 · Distributed and Cloud Computing

Remote Invocation

Request–Reply Protocol · Failure Handling · Exchange Protocols · HTTP Methods · RPC · RMI

1. Request–Reply Protocols

A pattern on top of message passing supporting two-way exchange in client–server computing. Synchronous: the client blocks until the reply arrives. Reliable: the reply is effectively an acknowledgement. Built over UDP rather than TCP because acknowledgements are redundant, TCP connection setup adds overhead, and flow control is redundant for small arguments.

Name	*Messages sent by*		
	Client	*Server*	*Client*
R	*Request*		
RR	*Request*	*Reply*	
RRA	*Request*	*Reply*	*Acknowledge reply*

Figure 1.1 Request–reply protocol: client blocks until reply arrives from server.

Message Structure

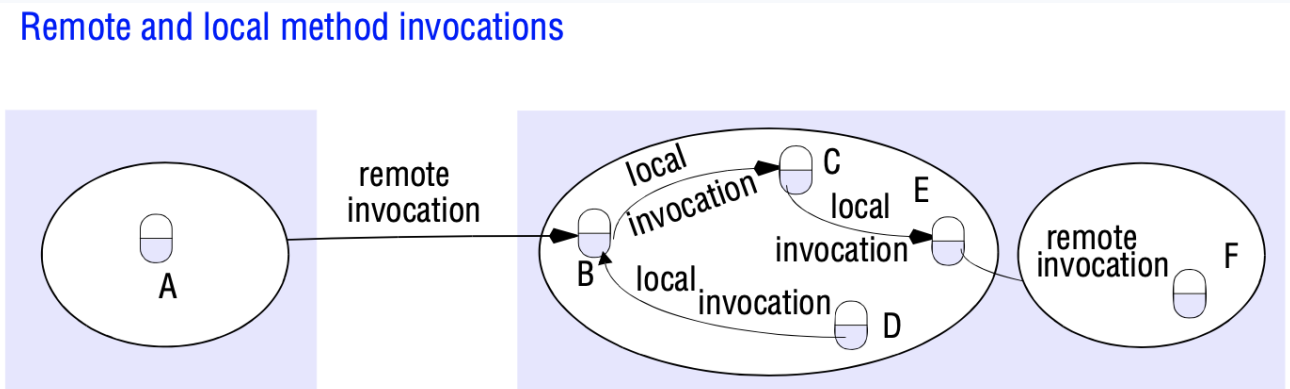

Figure 1.2 Structure of a request–reply message: messageType, requestId, remoteReference, operationId.

Field	Description
messageType	Indicates whether the message is a Request or a Reply.

requestId	Unique identifier with two parts: (1) increasing sequence of integers (unique to sender, reset at $2^{32}-1$); (2) sender process identifier (port + IP address, unique in the distributed system).
remoteReference	Identifies the target remote server object: Internet address and port.
operationId	Identifier for the operation to be invoked (number or reflection-based name).

1.1 Failure Handling

- **Omission failures:** messages may be dropped (checksum error or lack of buffer space).
- **Timeouts:** doOperation retransmits repeatedly until reply received or server assumed failed.
- **Duplicate filtering:** recognises successive messages with same requestId from same client.
- **Idempotent operations:** can be performed repeatedly with the same effect (e.g. add to set). Non-idempotent: append to sequence (extends it each time).
- **History:** structure storing transmitted reply messages (requestId, message, clientId) for retransmission without re-execution. Memory cost is a problem.

1.2 Exchange Protocols

- **R – Request:** single Request message; no reply; client proceeds immediately; used when no confirmation is needed.
- **RR – Request–Reply:** most common; server reply IS the acknowledgement; failures masked by retransmission + duplicate filtering + history.
- **RRA – Request–Reply–Acknowledge Reply:** three messages; Acknowledge reply enables server to discard history entries.

Fault tolerance measures			*Call semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	*Maybe*
Yes	No	Re-execute procedure	*At-least-once*
Yes	Yes	Retransmit reply	*At-most-once*

Figure 1.3 Exchange protocols: R (request only), RR (request–reply), and RRA (with acknowledgement).

2. HTTP — An Example Request–Reply Protocol

Implemented over TCP. Original version: connect → request → reply → close.

Method	Description
GET	Retrieve resource at given URL; no side effects.
HEAD	Same as GET but returns only the status line and headers, not data.
POST	Send data to server; may change server state (form submission, mailing list, database append).
PUT	Store data at given URL; modification of existing or new resource.
DELETE	Server deletes resource at given URL.
CONNECT	Establish network connection to server (HTTP tunnelling via proxy).
OPTIONS	Server returns list of methods allowed for given URL.
TRACE	Server sends back the request message; used for diagnostics.

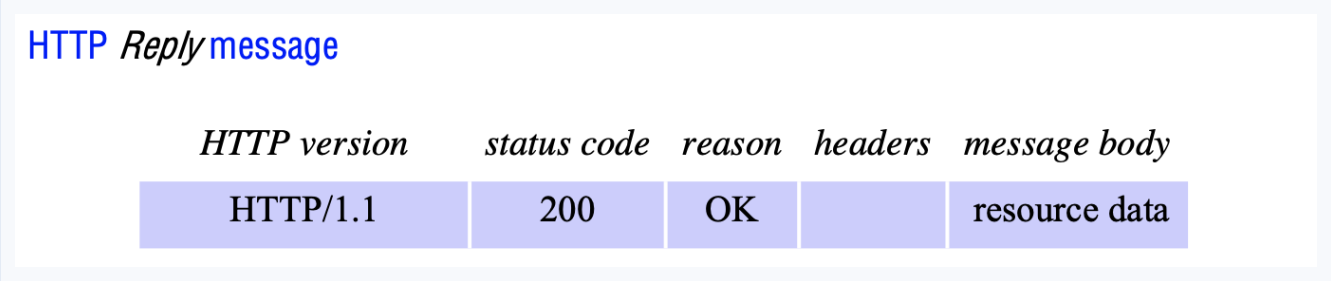

Figure 2.1 HTTP request message structure: method, URL, version, headers, and optional body.

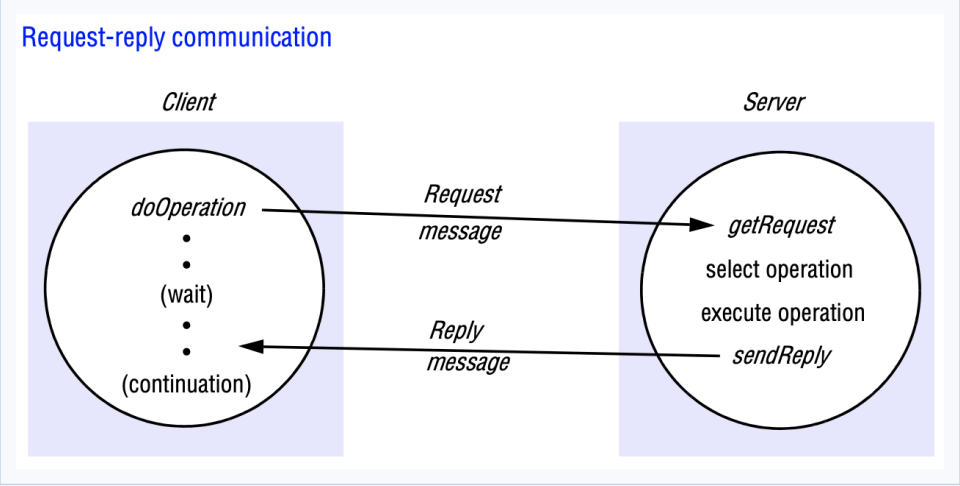

Figure 2.2 HTTP response message structure: status line, headers, and response body.

HTTP *Request* message

method	*URL or pathname*	*HTTP version*	*headers*	*message body*
GET	http://www.dcs.qmul.ac.uk/index.html	HTTP/ 1.1		

Figure 2.3 HTTP status code ranges: 1xx informational, 2xx success, 3xx redirect, 4xx client error, 5xx server error.

3. Remote Procedure Calls (RPC)

Procedures on remote machines called as if local. Hides: encoding/decoding parameters, message passing, and call semantics.

Interfaces in Distributed Systems

- The service interface specifies the procedures offered by a server and defines their argument types.
- Parameters are described as input or output, not by value/reference (incompatible across processes).
- Addresses in one process are not valid in another — cannot be passed as arguments.

RPC Call Semantics

Semantics	Behaviour	Appropriate Use
Maybe	Executed once or not at all; no fault tolerance.	When occasional failures are acceptable.
At-least-once	Retransmission masks omission; may execute more than once.	Idempotent operations only.
At-most-once	Retries + duplicate filtering; executed exactly once or not at all.	Non-idempotent operations (the general case).

Implementation

- **Client stub:** marshals procedure ID and arguments; sends; unmarshals results.
- **Dispatcher:** selects server stub by procedure identifier.
- **Server stub:** unmarshals arguments; calls service procedure; marshals return values.

Stubs and dispatcher are generated automatically by an interface compiler.

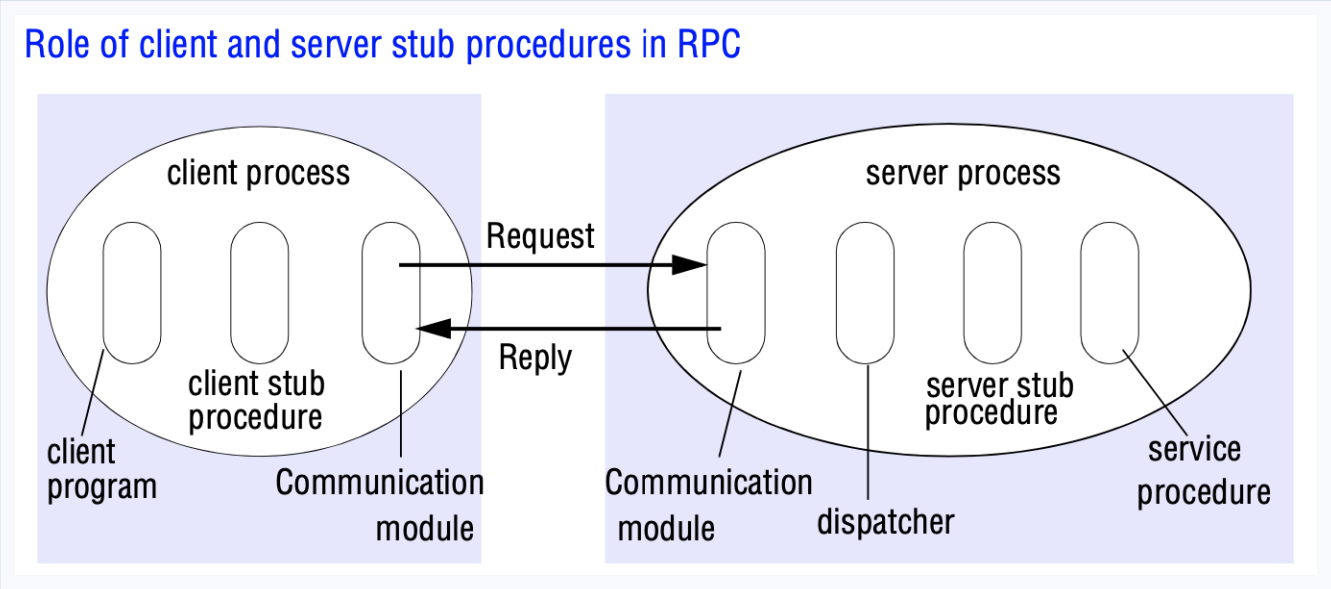

Figure 3.1 RPC implementation: client stub, dispatcher, server stub, and service procedures.

4. Remote Method Invocation (RMI)

RPC extended to distributed objects: a calling object invokes a method on a potentially remote object. Same benefits as RPC (interfaces, at-most-once, transparency), plus: full OOP expressiveness (objects, classes, inheritance) and unique object references that can be passed as parameters.

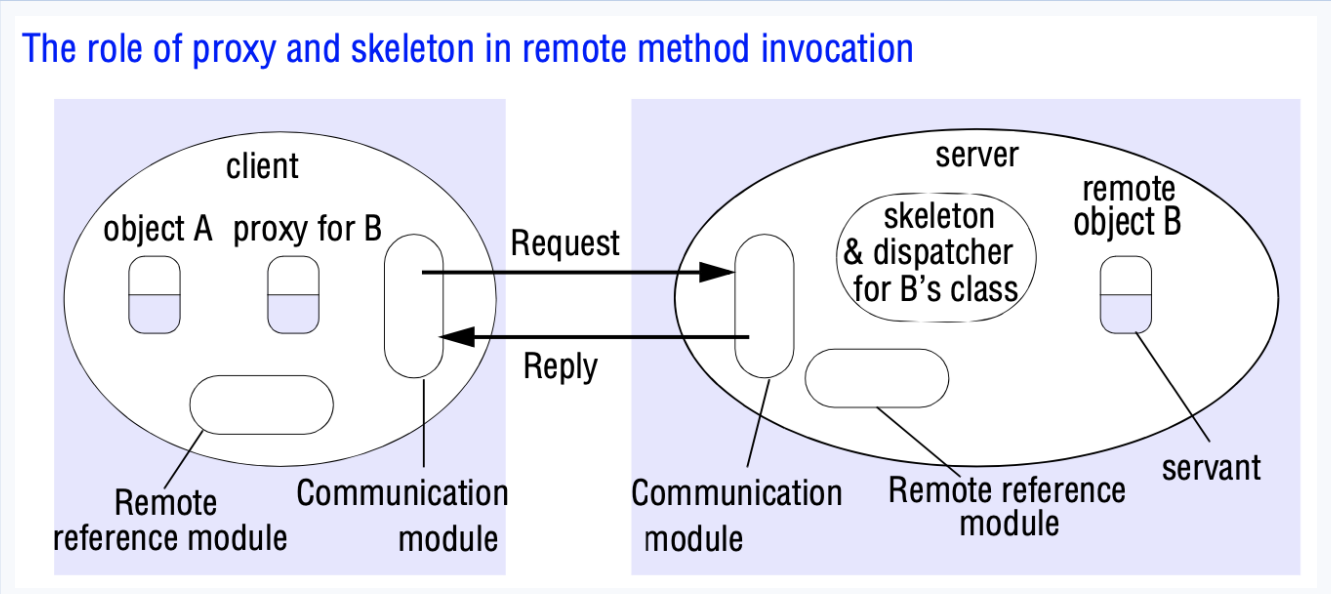

Figure 4.1 RMI architecture: proxy, dispatcher, skeleton, servant, and remote reference module.

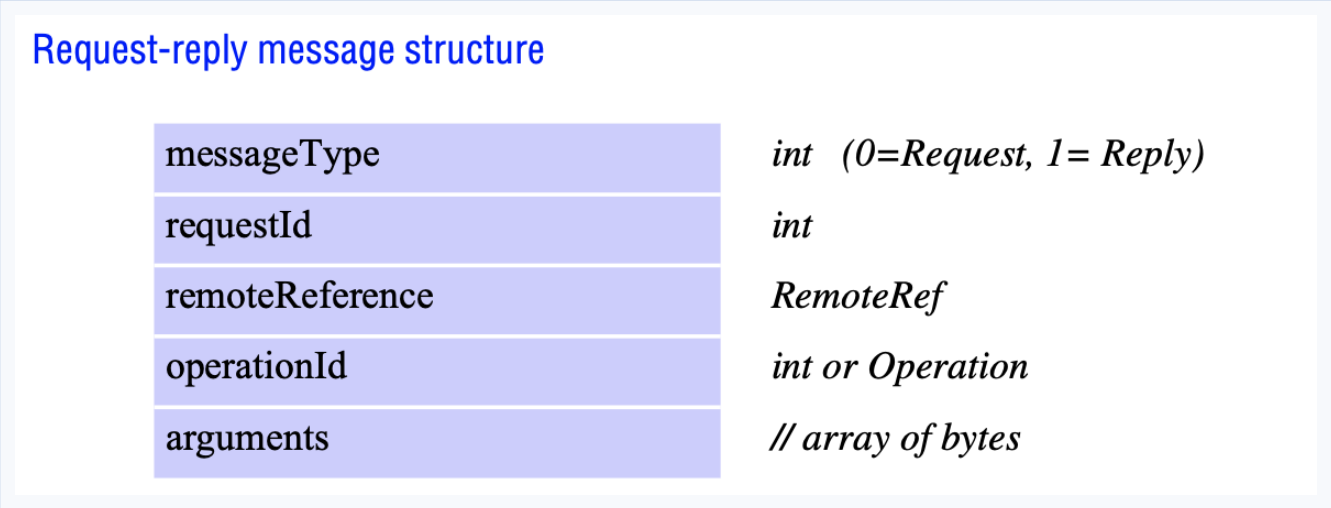

Figure 4.2 Distributed object model: objects B and F have remote interfaces and remote object references.

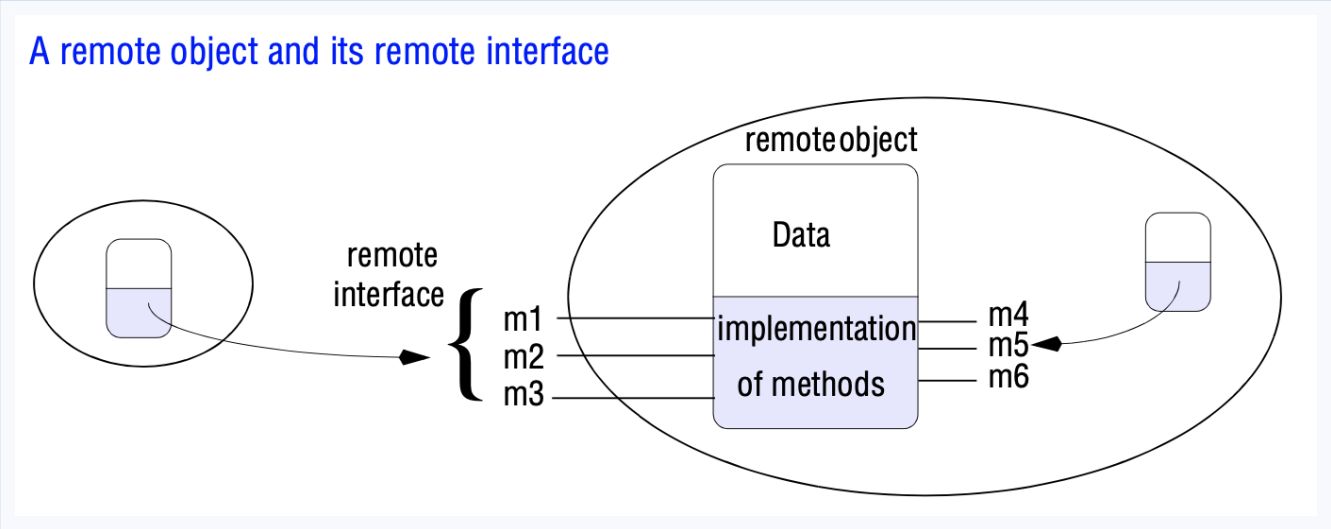

Figure 4.3 RMI component interactions: client proxy, communication module, remote reference module, skeleton.

Exercises

- 1. Design a HelloWorld system incorporating Java RPC. Publish a Medium article. Ref: javatpoint.com/jax-ws-example-rpc-style
- 2. Try out the Java RMI example and explain the concepts. Publish a Medium article. Ref: javatpoint.com/RMI