Distributed Systems

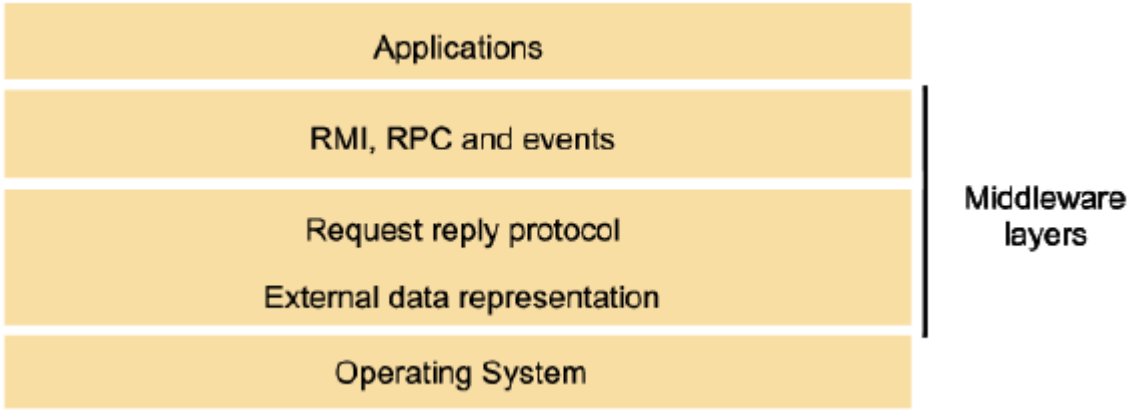
COMP90015 2018 Semester 1
Tutorial 06

Today’s Agenda

- Questions of Remote Invocation
- Demonstration of Remote Method Invocation (RMI)

Remote Invocation

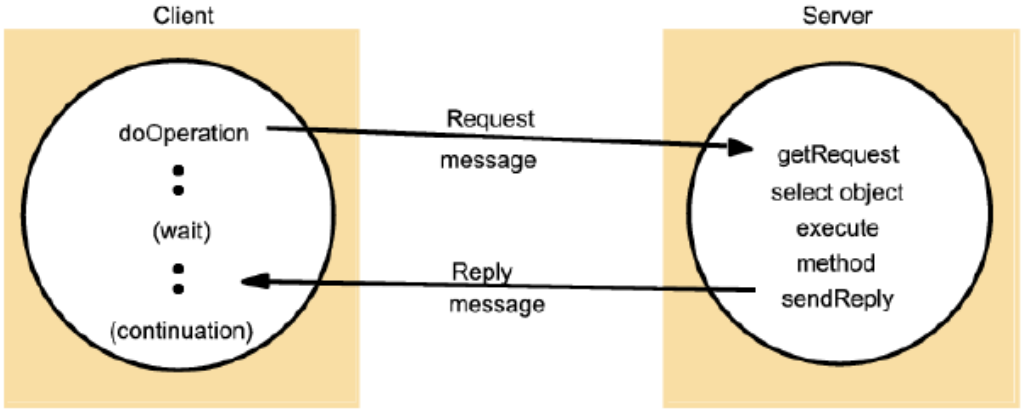
- A set of information exchange protocols at the Middleware layer



- Can you define and explain the procedure of the Request-Reply protocol?

Request-Reply protocol

- Can you describe the procedure of the Request-Reply protocol?



Possible Issues in Request-Reply

- What are the issues that may arise in a request-reply process?
  - Request timeouts
    - Retry request message after timeout
    - Do nothing
  - Reply timeouts
    - Retransmit results after timeout
    - Do nothing
  - Receiving duplicate requests/replies
    - Discard duplicate messages
    - Perform the same action again if the logic is idempotent

Invocation Semantics

- Different issue handling strategies lead to different invocation semantics

- Invocation semantics
  - Define what the client can assume about the execution of the remote procedure
  - Offer different reliability guarantees in terms of the number of times that the remote procedure is executed

- Classification of Request/Reply protocols based on invocation semantics
  - The request protocol (R)
  - The request-reply protocol (RR)
  - The request-reply-acknowledge protocol (RRA)

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

Can you explain the following invocation semantics?

- *Maybe semantics*
- *At-least-once semantics*
- *At-most-once semantics*

Different Invocation Semantics

- **Maybe**:
  - The remote procedure call may be executed once or not at all.
  - Unless the caller receives a result, it is unknown as to whether the remote procedure was called.
  - Suitable for applications in which occasional failed calls are acceptable
- **At-least-once**:
  - Either the remote procedure was executed at least once, and the caller received a response, or
  - The caller received an exception to indicate the remote procedure was not executed at all.
  - Suitable for idempotent operations.
- **At-most-once**:
  - The remote procedure call was either executed exactly once, in which case the caller received a response, or
  - It was not executed at all and the caller receives an exception.
  - Suitable for non-idempotent operations.

Figure 5.5
Invocation semantics

Fault tolerance measures			*Invocation 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*

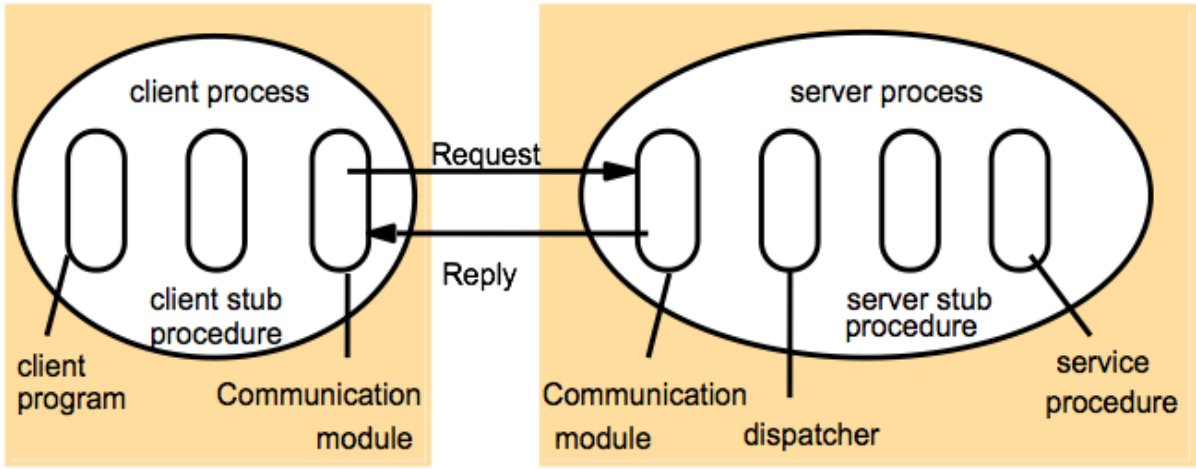
Remote Procedure Call

- Please explain Remote procedure call (RPC) and describe its key components.
    - RPCs enable clients to execute procedures in server processes based on a defined service interface.
    - Used in procedural languages such as Fortran, C, and GO.

Remote Procedure Call (RPC) and key components

Key components of RPC:

- Communication Module
- Client Stub Procedure
- Dispatcher
- Server stub procedure

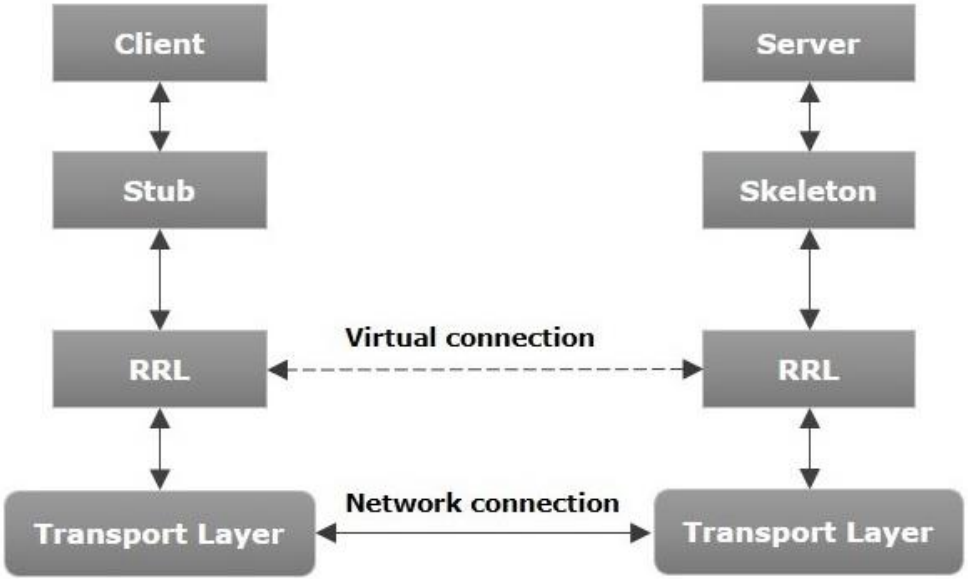


Remote procedure call (RPC) and key components

- **Communication Module**
  Implements the desired design choices in terms of retransmission of requests, dealing with duplicates and retransmission of results
- **Client Stub Procedure**
  Behaves like a local procedure to the client. Marshals the procedure identifiers and arguments which is handed to the communication module
  Unmarshalls the results in the reply
- **Dispatcher**
  Selects the server stub based on the procedure identifier and forwards the request to the server stub
- **Server Stub Procedure**
  Unmarshalls the arguments in the request message and forwards it to the Service Procedure. Marshalls the arguments in the result message and returns it to the client

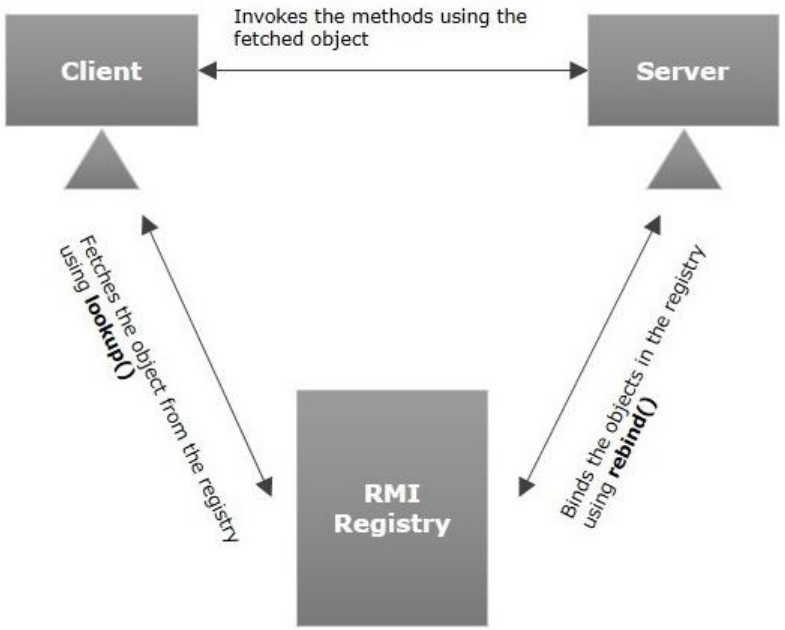
Java RMI

- Explain the steps involved in Java RMI to build a distributed system.

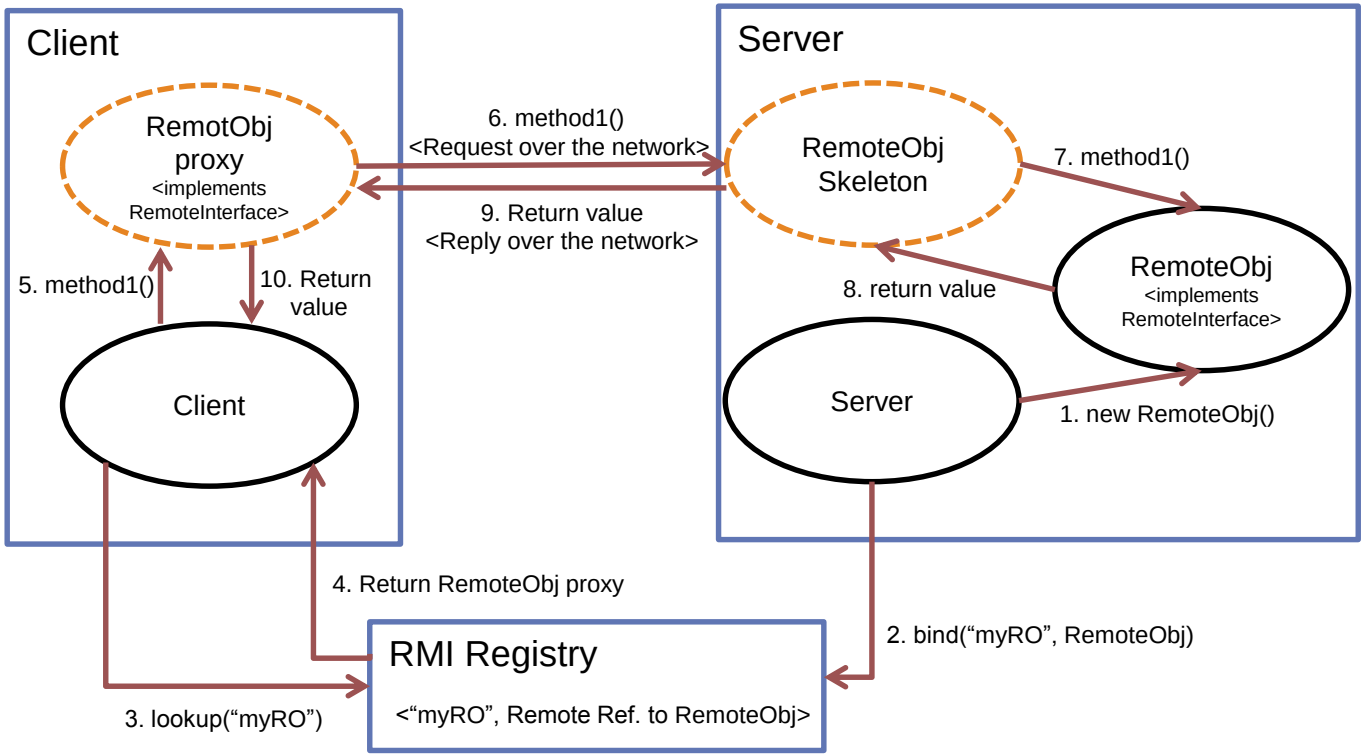


Java RMI

- Explain the steps involved in Java RMI to build a distributed system.



Java RMI Overview



Explain the steps involved in Java RMI to build a distributed system.

- Define the remote interface
  - Defines the methods that can be remotely invoked
  - Extends java.rmi.Remote interface
  - All methods throw java.rmi.RemoteException
  - Needs to be included in the client and in the server programs
- Server program
  - Create the remote object
  - Implements the remote interface
  - Extends UnicastRemoteObject
  - Implement the actual server
    - Export the remote object into the Java RMI runtime so that it can receive remote calls
    - Locate the RMIregistry and publish the remote object on it (the registry stores <name, remoteObjRef> pairs.
- Client program
  - Locate the RMIregistry and lookup the remote object by name
  - Invoke methods on the remote object as if it was a local one

Demo Time!

RMI Demo