Distributed Systems

COMP90015 2021 Semester 1
Tutorial 07

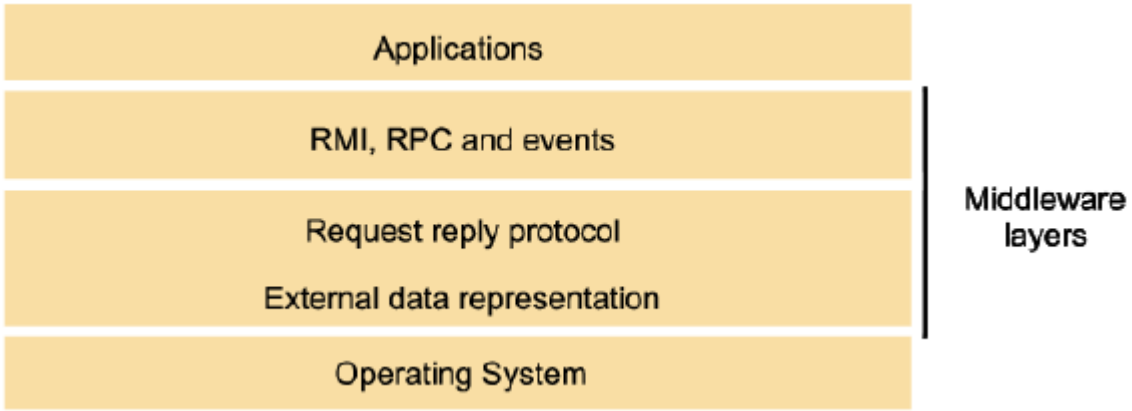
Today’s Agenda

1. Questions/Discussion on Remote Invocation, RPC and RMI
2. Code Demonstration : RMI Client Server

What is Remote Invocation?

Remote Invocation

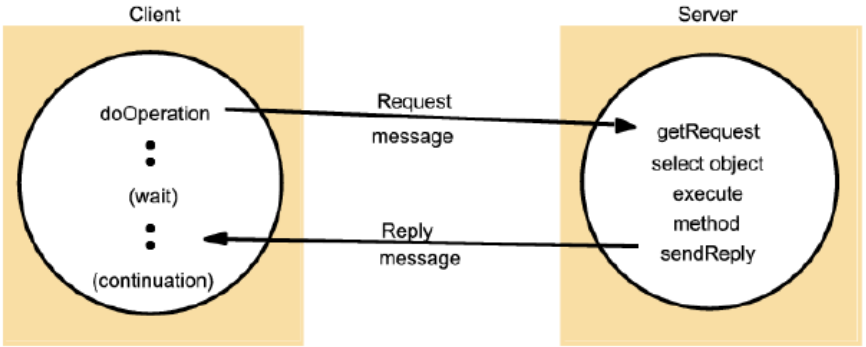
- A set of information exchange protocols at the Middleware layer



1. Can you explain three different types of protocols typically used to address the design issues of Remote Invocation?

1. Can you explain three different types of protocols typically used to address the design issues of Remote Invocation?

- **Request** : Client sends message
- **Request-Reply** : Client sends message, Server sends reply
- **Request-Reply-Acknowledgement** : Same as above, but once the client receives the message, the client will acknowledge it.



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

2. Explain different invocation semantics

2. Explain different 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

2. Explain 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.
- **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*

3. Please explain Remote procedure call (RPC) and describe its key components.

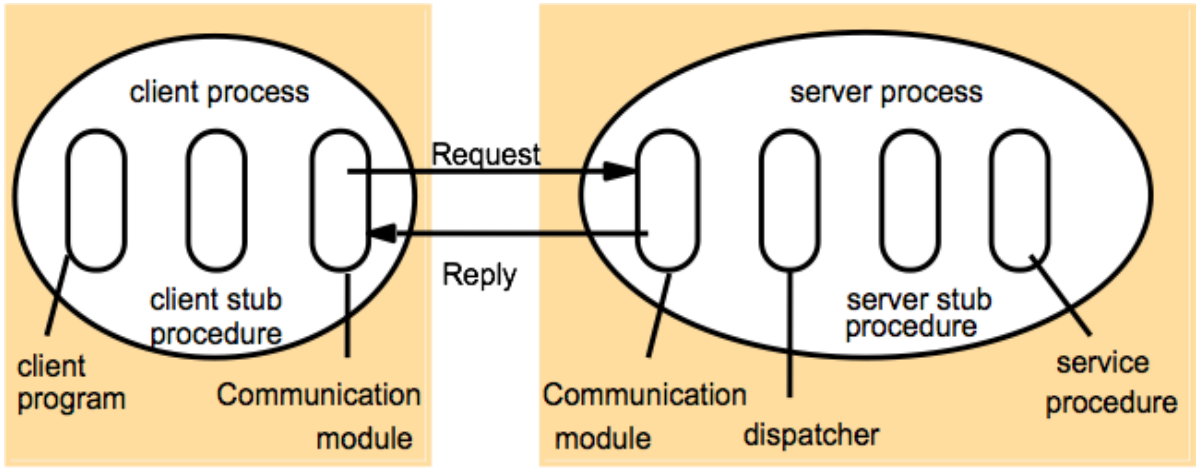
3. Remote Procedure Call

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

3. Remote Procedure Call (RPC) and key components

Key components of RPC:

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

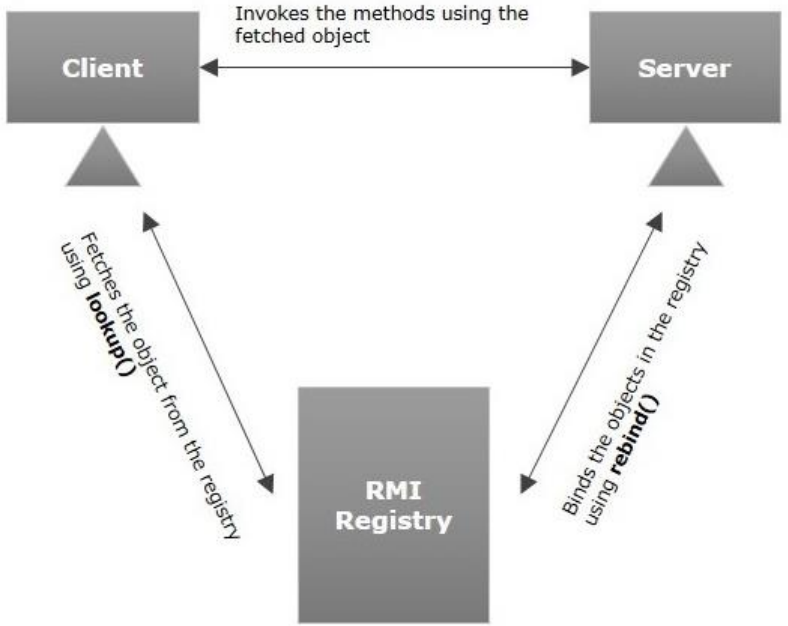


3. Remote procedure call (RPC) and key components3.

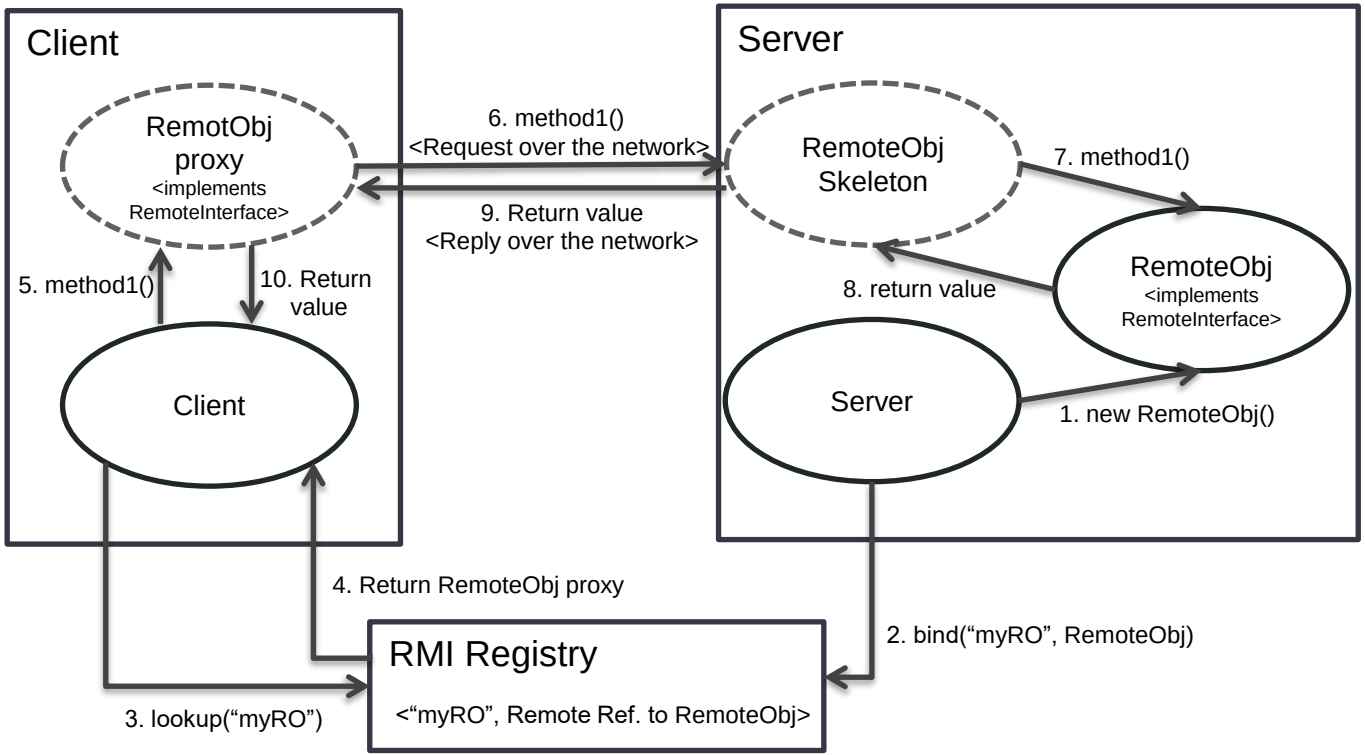
- **Communication Module**
  Implements the desired design choices in terms of retransmission of requests, dealing with duplicates and retransmission of results
- **Client Stub Procedure**
  Behaves like a local procedure to the client. Marshals the procedure identifiers and arguments which is handed 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

4. Explain how one goes about using Java RMI to build a distributed system. Discuss what needs to be compiled and important aspects of the system that must be used. Actual code is not required.

Java RMI


Client
Invokes the methods using the fetched object
Server
Fetches the object from the registry using lookup()
RMI Registry
Binds the objects in the registry using rebind()

Java RMI Overview



4. Explain how one goes about using Java RMI to build a distributed system. Discuss what needs to be compiled and important aspects of the system that must be used. Actual code is not required.

- *Defining the interface for remote objects* - Interface is defined using the interface definition mechanism supported by the particular RMI software.
- *Compiling the interface* - Compiling the interface generates proxy, dispatcher and skeleton classes.
- *Writing the server program* - The remote object classes are implemented and compiled with the classes for the dispatchers and skeletons. The server is also responsible for creating and initializing the objects and registering with the binder.
- *Writing client programs* - Client programs implement invoking code and contain proxies for all remote classes. Uses a binder to lookup for remote objects.

Demo Time!

RMI Demo