UNIVERSITAT ROVIRA i VIRGILI

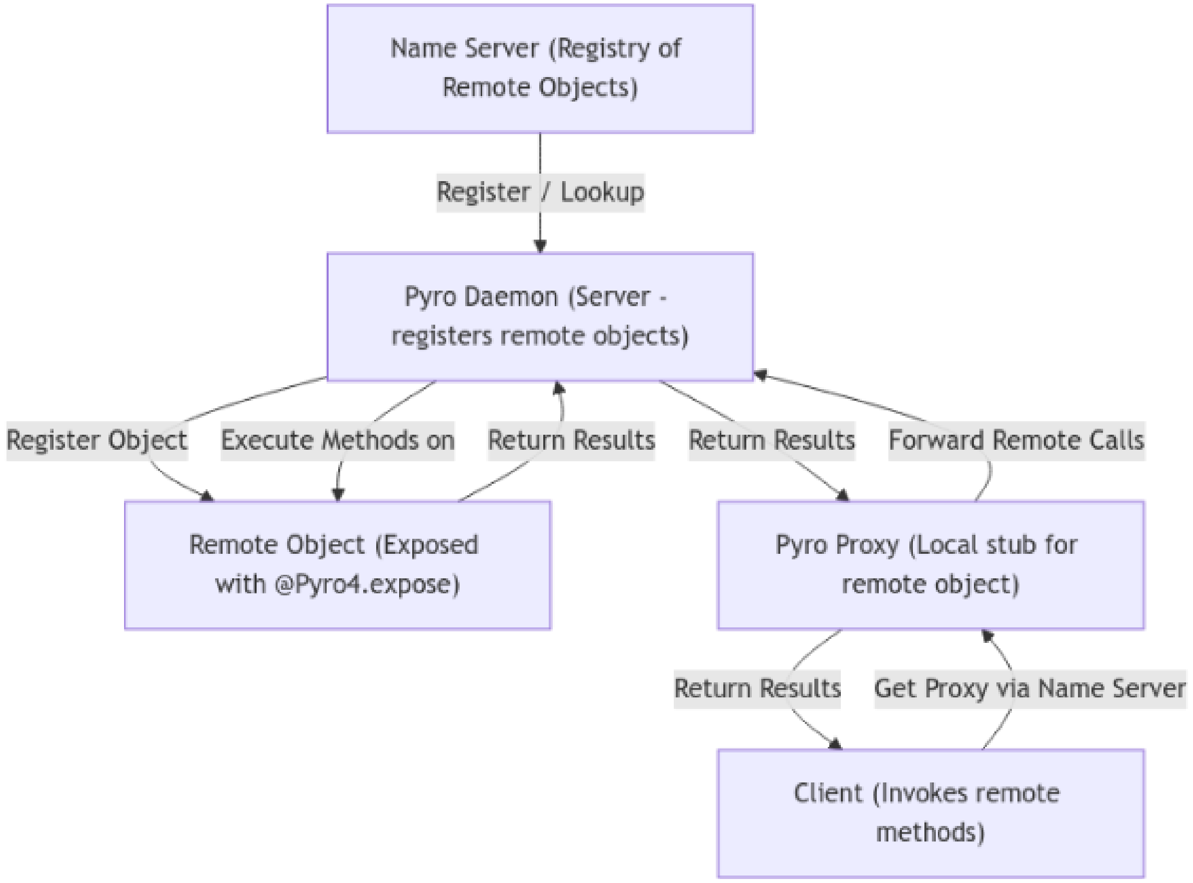
Lab 2 – Pyro4

Distributed Systems

Introduction to Pyro4

- **What is Pyro4?**
  - A Python framework for remote object communication.
  - Allows you to expose objects and methods across processes or machines.
  - Simplifies the development of distributed applications.
- **Key components**
  - **Daemon:** Listens for and handles remote method calls.
  - **Proxy:** A local representative (stub) that communicates with the remote object.
  - **Name Server:** A central registry for naming and locating remote objects.

Pyro4 Architecture


Name Server (Registry of Remote Objects)
Register / Lookup
Pyro Daemon (Server - registers remote objects)
Register Object
Execute Methods on
Return Results
Return Results
Forward Remote Calls
Remote Object (Exposed with @Pyro4.expose)
Pyro Proxy (Local stub for remote object)
Return Results
Get Proxy via Name Server
Client (Invokes remote methods)

Pyro4 Workflow

- How it works?
  - **Name Server Startup:**
    Start the Name Server (typically via `python -m Pyro4.naming`) to register remote objects.
  - **Server Registration:**
    The server creates a Pyro Daemon, registers its objects, and then registers them with the Name Server under a specific name.
  - **Client Connection:**
    The client uses a Proxy (e.g., `"PYRONAME:echo.server"`) to locate the remote object via the Name Server.
  - **Remote Invocation:**
    The client calls methods on the Proxy, which are forwarded to the remote object.
- Benefits
  - Transparent remote method invocation.
  - Decoupling of object location and communication.

Exercise 1 – EchoServer

- **Server Code**
  - The `EchoServer` class is exposed with `@Pyro4.expose`.
  - The server creates a Pyro Daemon, locates the Name Server, registers the object, and then registers it with the Name Server using the name `"echo.server"`.
  - The code then enters the request loop to wait for incoming requests.
- **Client Code**
  - The client sets the configuration for the Name Server (host and port).
  - It retrieves the remote object using `"PYRONAME:echo.server"` and calls `echo("HOLA")`.
  - The response is printed.

Exercise 2 – Observer Pattern

- **Observable Server:**
  - The `Observable` class is defined and exposed using both `@Pyro4.expose` and `@Pyro4.behavior(instance_mode="single")`.
  - It contains methods to register and unregister observers, as well as to notify them.
  - The server registers the Observable object with the Name Server under `"example.observable"` and enters the request loop.
- **Observer Client:**
  - The `Observer` class is defined with an `update` method that prints received messages.
  - The observer script locates the Name Server, retrieves the Observable object (using `ns.lookup("example.observable")`), and registers itself with the observable using its remote URI.
  - The observer then waits in a request loop to receive notifications.

Exercise 2 – Observer Pattern

- **Notification Script:**
  - The separate script to notify observers creates a proxy for `"PYRONAME:example.observable"` and calls `notify_observers("Hello, Observers!")`.

Exercise 3 – MyRemoteObject & Dynamic Introspection

- **Server Code**
  - The `MyRemoteObject` class is defined with methods `greet` and `add`.
  - The class is exposed with `@Pyro4.expose` and registered with the Name Server under `"example.remote.object"`.
  - The server prints the URI and enters the request loop.
- **Client Code**
  - The client locates the Name Server, looks up the object `"example.remote.object"`, and creates a proxy.
  - It calls both the `greet` and `add` methods, prints their results, and then performs dynamic introspection by listing `_pyroMethods`.

Running the Exercises

- **Step 1**
  - Start the Pyro Name Server on the designated port to enable remote object registration and lookup.
- **Step 2**
  - Launch the server application for the specific exercise (e.g., the EchoServer, Observable, or MyRemoteObject server) so that the remote objects are registered and available.
- **Step 3**
  - In separate terminals, run the client, observer, or notifier applications that connect to the Name Server to interact with the remote objects.
- **Step 4**
  - Verify that the system is working correctly by checking the console outputs in each terminal for the expected responses and notifications.

Lab 2 assignment

- **Resources:**
  - Official Documentation: Pyro4 Documentation
  - GitHub Repository: Pyro4 on GitHub

UNIVERSITAT
ROVIRA i VIRGILI