De La Salle University
Computer Technology Department

NSDSYST

Members:

Group No.: ________________

Date: ________________

Hands-On 1 – Remote Objects on Python Using Pyro 4

1 Introduction

A remote object is an object that defines methods that can be called by a client located in a remote node or computer. Remote objects are implemented by having a remote interface that declared remote methods of an object. Normally, the object can be located using its uniform resource identifier. Python allows invoking of remote methods in objects using the Pyro library. The Pyro library allows normal method calls on the client that executes on a “server” node. Python Pyro uses its own nameserver (called pyro-ns) to allow nodes to register their objects which allows finding of node with a certain method be “faster”.

2 Objectives

Objectives for the laboratory session are:

- Understand how remote objects work in distributed system
- Create a server that executes code for client

3 Requirements

- Ubuntu Desktop with Virt-Manager or Virtual-Box
- Ubuntu Server virtual machine

4 Lab Procedures

1. Install the Pyro4 library for python, by typing in “`pip3 install pyro4`”. For windows machine, type in the command “`pip install pyro4`”.
2. In the command line, the Pyro4 name server has to be running. To run the name server, type in the command “`pyro4-ns -h w.x.y.z`”. The `w.x.y.z` is the ip address of the machine running the name server.
3. On another terminal window, type in the command “`pyro4-nsc list`”. This will show the list of the remote objects registered on the name server.
4. Type in the following the program for the server:

```
import Pyro4, random

motd=["Fortune and love favor the brave.", "Live as brave men; and if
fortune is adverse, front its blows with brave hearts.", "Behind every great
fortune lies a great crime."]

```

```
@Pyro4.expose
class MotdMaker(object):
    def __init__(self):
        self.msg=motd[random.randint(0,2)]
    def get_motd(self, name):
        return "Message of the day for {0}:\n {1}".format(name, self.msg)

daemon = Pyro4.Daemon()
uri = daemon.register(MotdMaker)

print("Ready. Object uri =", uri)
daemon.requestLoop()
```

5. Run the server program and copy the URI of the object.

6. Type in the following the program for the client:

```
import Pyro4

uri = input("What is the Pyro uri of the greeting object? ").strip()
name = input("What is your name? ").strip()

motd_maker = Pyro4.Proxy(uri)
print(motd_maker.get_motd(name))
```

7. Before running the client program, type in the command “pyro4-nsc list”. Do you see the URI of the object in the server?

8. Run the client program and type in the URI of the object in the server and other information when prompted. Describe the output of the code.

9. Was the object code executed in the client? Why?

10. So far, all the codes are run on the same computer. Run the server code below that allows remote connection.

```
import Pyro4, random

motd=["Fortune and love favor the brave.", "Live as brave men; and if
fortune is adverse, front its blows with brave hearts.", "Behind every great
fortune lies a great crime."]
```

```

@Pyro4.expose

class MotdMaker(object):
    def __init__(self):
        self.msg=motd[random.randint(0,2)]
    def get_motd(self, name):
        print ("request made \n")
        return "Message of the day for {0}:\n{1}".format(name, self.msg)

def main():
    Pyro4.Daemon.serveSimple(
            {
                MotdMaker: "example.motd"
            },
            host= 'w.x.y.z', #change this to the IP address of the computer
            ns = True)

if __name__=="__main__":
    main()
```

11. On the Ubuntu virtual machine, run the code client code below:

```
import Pyro4

name = input("What is your name? ").strip()

motd_maker = Pyro4.Proxy("PYRONAME:example.motd")# use name server object
lookup uri shortcut
msg=motd_maker.get_motd(name)
print(msg)
```

12. Was the object code executed in the remote server or client? Why?

13. Was the URI used to locate the object on the server? How was the object located? Explain.

5 Reflection / Questions

1. What are remote objects?

2. How can remote objects be invoked on another computer?

3. What are the applications of remote objects?

4. How can locks be used to implement mutual exclusion among threads when using shared variables in an application?