PARALLEL AND DISTRIBUTED SYSTEMS
FINAL PROJECT: RPC VERSIONING

Lecturer:
Siti Amatullah Karimah, S.T., M.T.

Group 1:

Rexcy Putra Adrianto	1301201589
Shinta Dewi L.S	1301203567
Olaza Aurora Syafira	1301202610
Nur Afina Rahmani	1301202563
M. Fadli Ramadhan	1301203533

IF-44-INT

School of Computing – Informatics
Telkom University

1. Introduction

Versioning

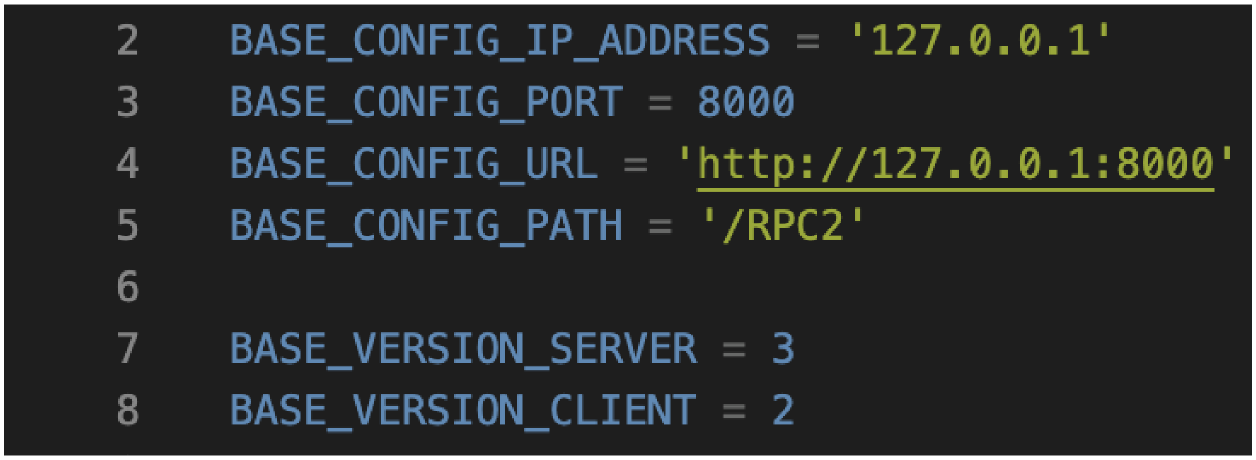
Implement Simple Versioning Control in Python

About This Project

To make a simple program for the server to give the information to the client about the versioning/application information update. If the client do the updating, then the application in the client will be changed based on its update.

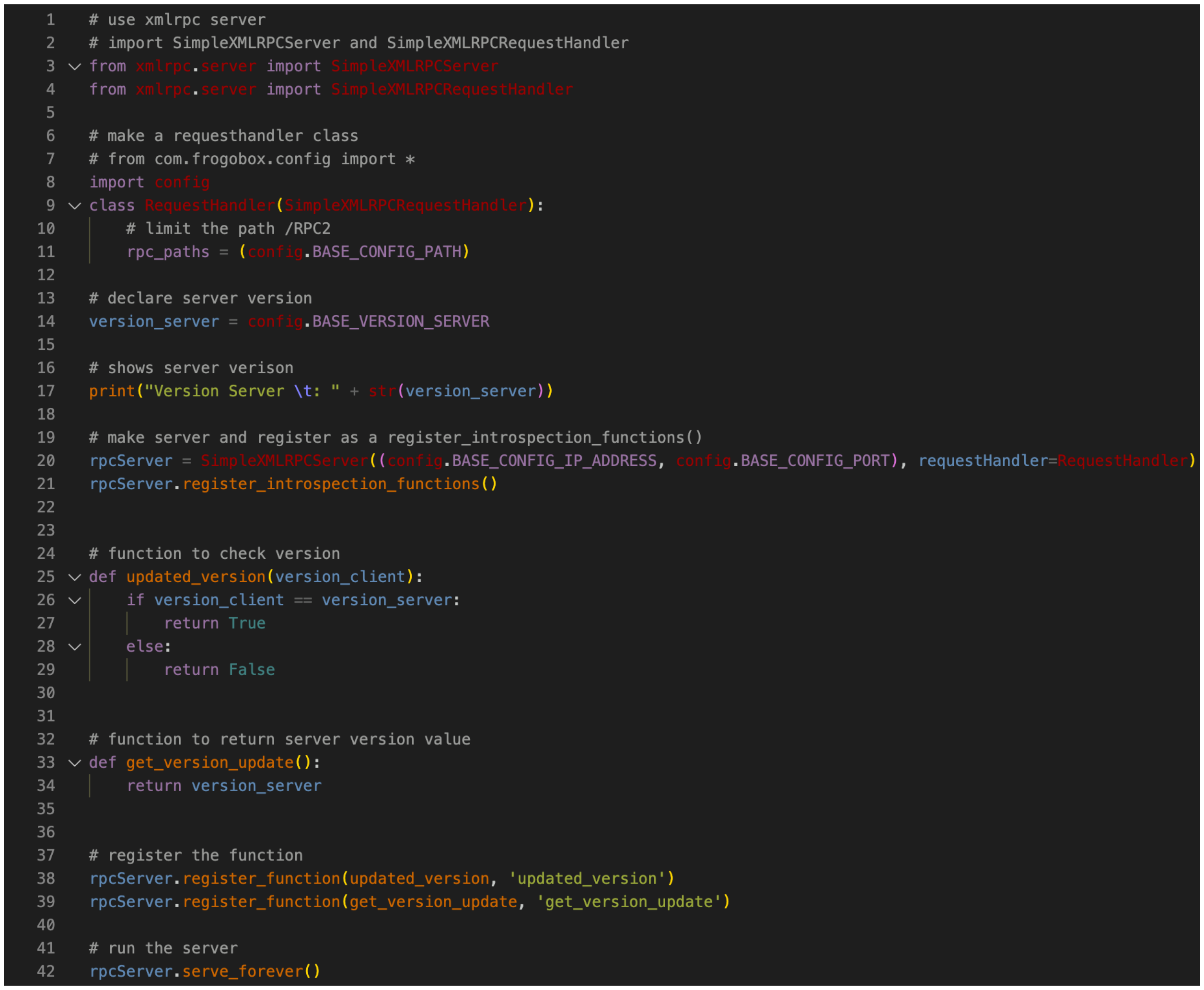
2. Result

config.py



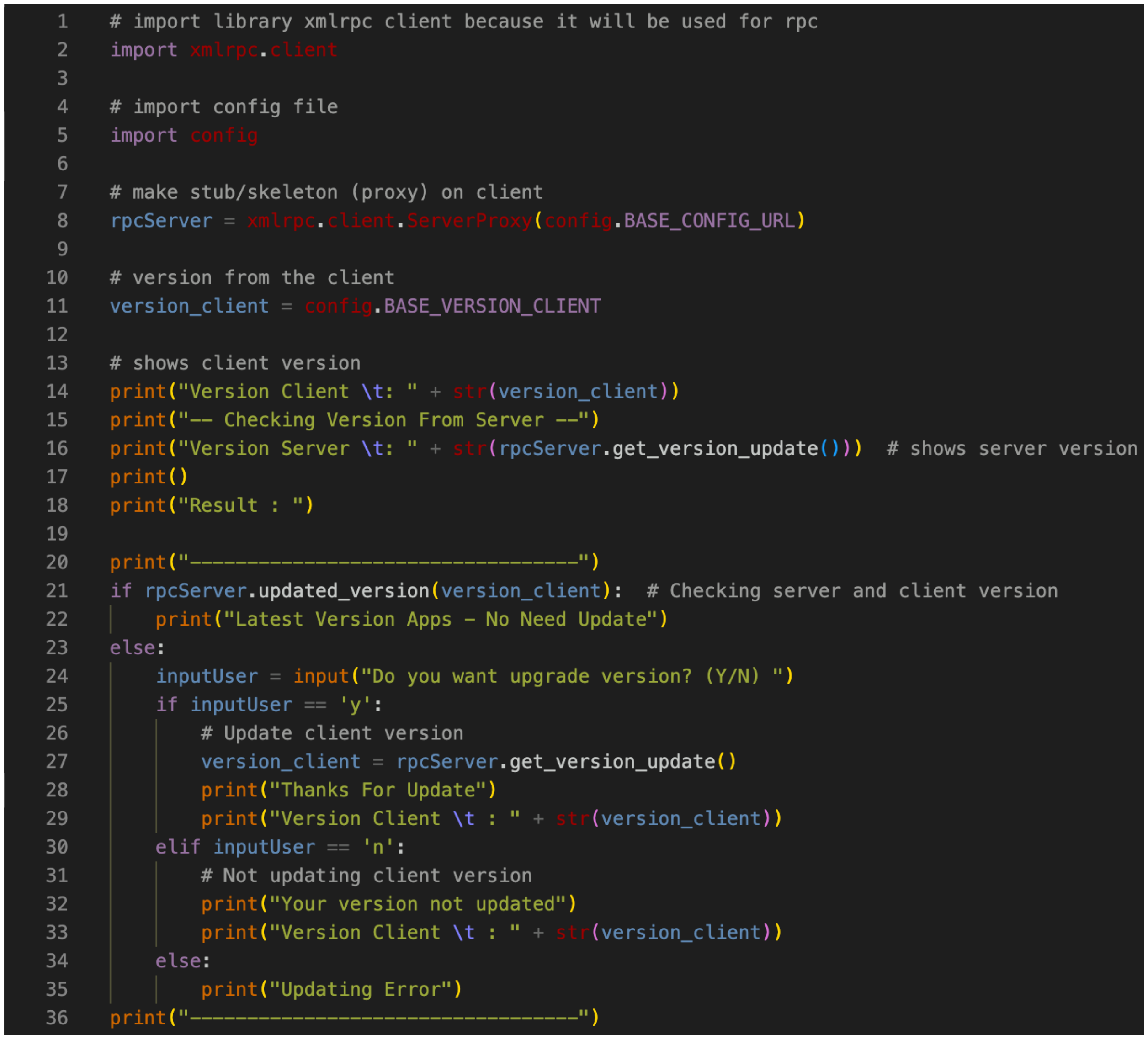
```python
BASE_CONFIG_IP_ADDRESS = '127.0.0.1'
BASE_CONFIG_PORT = 8000
BASE_CONFIG_URL = 'http://127.0.0.1:8000'
BASE_CONFIG_PATH = '/RPC2'

BASE_VERSION_SERVER = 3
BASE_VERSION_CLIENT = 2
```

rpc_client.py



```python
# use xmlrpc server
# import SimpleXMLRPCServer and SimpleXMLRPCRequestHandler
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler

# make a requesthandler class
# from com.frogobox.config import *
import config
class RequestHandler(SimpleXMLRPCRequestHandler):
    # limit the path /RPC2
    rpc_paths = (config.BASE_CONFIG_PATH)

# declare server version
version_server = config.BASE_VERSION_SERVER

# shows server verison
print("Version Server \t: " + str(version_server))

# make server and register as a register_introspection_functions()
rpcServer = SimpleXMLRPCServer((config.BASE_CONFIG_IP_ADDRESS, config.BASE_CONFIG_PORT), requestHandler=RequestHandler)
rpcServer.register_introspection_functions()


# function to check version
def updated_version(version_client):
    if version_client == version_server:
        return True
    else:
        return False


# function to return server version value
def get_version_update():
    return version_server


# register the function
rpcServer.register_function(updated_version, 'updated_version')
rpcServer.register_function(get_version_update, 'get_version_update')

# run the server
rpcServer.serve_forever()
```

rpc_server.py



```python
# import library xmlrpc client because it will be used for rpc
import xmlrpc.client

# import config file
import config

# make stub/skeleton (proxy) on client
rpcServer = xmlrpc.client.ServerProxy(config.BASE_CONFIG_URL)

# version from the client
version_client = config.BASE_VERSION_CLIENT

# shows client version
print("Version Client \t: " + str(version_client))
print("-- Checking Version From Server --")
print("Version Server \t: " + str(rpcServer.get_version_update()))  # shows server version
print()
print("Result : ")

print("-----------------------------------")
if rpcServer.updated_version(version_client):  # Checking server and client version
    print("Latest Version Apps - No Need Update")
else:
    inputUser = input("Do you want upgrade version? (Y/N) ")
    if inputUser == 'y':
        # Update client version
        version_client = rpcServer.get_version_update()
        print("Thanks For Update")
        print("Version Client \t : " + str(version_client))
    elif inputUser == 'n':
        # Not updating client version
        print("Your version not updated")
        print("Version Client \t : " + str(version_client))
    else:
        print("Updating Error")
print("-----------------------------------")
```

Code Analysis

RPC Server Analysis

Code	Description
`from xmlrpc.server import SimpleXMLRPCServer`	To make the server traverses the platform with its own language using the XML-RPC protocol
`from xmlrpc.server import SimpleXMLRPCRequestHandler`	To control new request

`import config`
`class RequestHandler(SimpleXMLRPCRequestHandler):`
	To make a new class for requesthandler

`version_server = BASE_VERSION_SERVER`	To make the version from the server

`rpcServer = SimpleXMLRPCServer((BASE_CONFIG_IP_ADDRESS,`
`BASE_CONFIG_PORT),`
`requestHandler=RequestHandler)`
`rpcServer.register_introspection_functions()`
	To make the introspection function

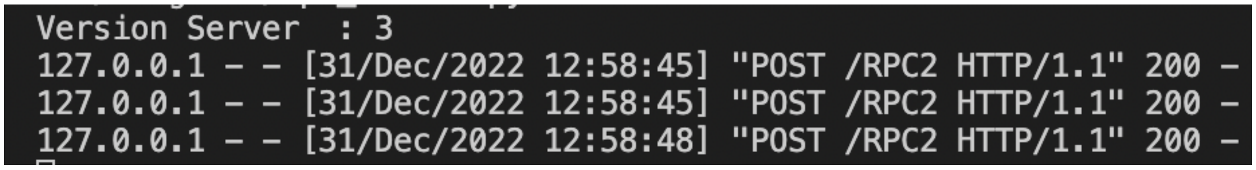
`def updated_version(version_client):`
`    if version_client == version_server:`
`        return True`
`    else:`
`        return False`
	Function that is used to check the version on the client and the server

`def get_version_update():`
`    return version_server`
	Function to return the value from the server

`rpcServer.register_function(updated_version,`
`'updated_version')`
`rpcServer.register_function(get_version_update,`
`'get_version_update')`
	To list the function so it can be used by the client

`rpcServer.serve_forever()`	To make the server can be used

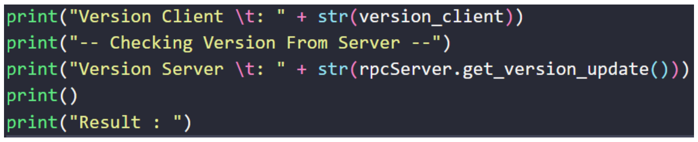
Server result

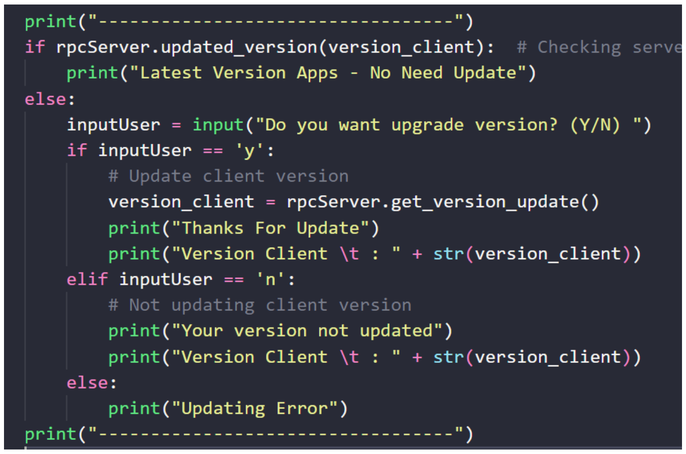


```
Version Server  : 3
127.0.0.1 - - [31/Dec/2022 12:58:45] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [31/Dec/2022 12:58:45] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [31/Dec/2022 12:58:48] "POST /RPC2 HTTP/1.1" 200 -
```

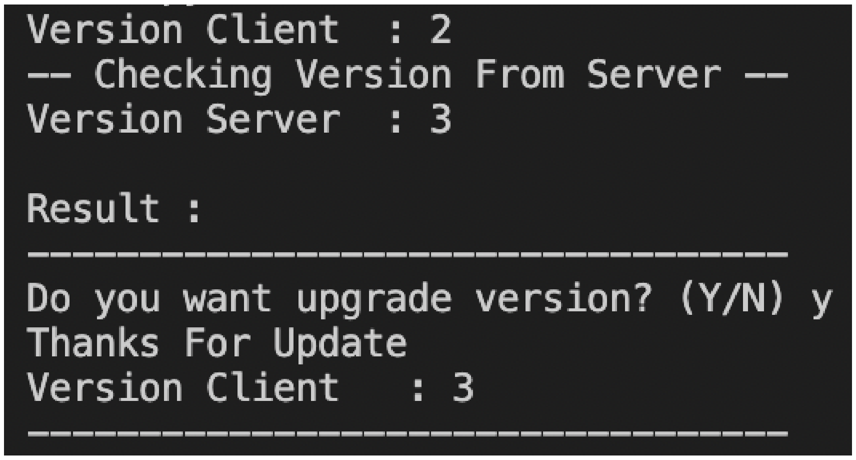
RPC Client Analysis

Code	Description
`import xmlrpc.client`	Calling the library that'll be used in the RPC.
`rpcServer = xmlrpc.client.ServerProxy(config.BASE_CONFIG_URL)`	The proxy for the client is made here.
`version_client = config.BASE_VERSION_CLIENT`	The variable client version

 `print("Version Client \t: " + str(version_client))`
`print("-- Checking Version From Server --")`
`print("Version Server \t: " + str(rpcServer.get_version_update()))`
`print()`
`print("Result : ")`
	Outputs the client's version and checks which version is this client.

 `print("----------------------------------")`
`if rpcServer.updated_version(version_client):  # Checking serve`
`    print("Latest Version Apps - No Need Update")`
`else:`
`    inputUser = input("Do you want upgrade version? (Y/N) ")`
`    if inputUser == 'y':`
`        # Update client version`
`        version_client = rpcServer.get_version_update()`
`        print("Thanks For Update")`
`        print("Version Client \t : " + str(version_client))`
`    elif inputUser == 'n':`
`        # Not updating client version`
`        print("Your version not updated")`
`        print("Version Client \t : " + str(version_client))`
`    else:`
`        print("Updating Error")`
`print("----------------------------------")`
	Checking the client's version. There will be 2 condition. First, where the it is the newest version which don't need an upgrade. Second, where it is not the newest version. User will then given the option to upgrade their version or not.

Client result



```
Version Client  : 2
-- Checking Version From Server --
Version Server  : 3

Result :
----------------------------------
Do you want upgrade version? (Y/N) y
Thanks For Update
Version Client   : 3
----------------------------------
```