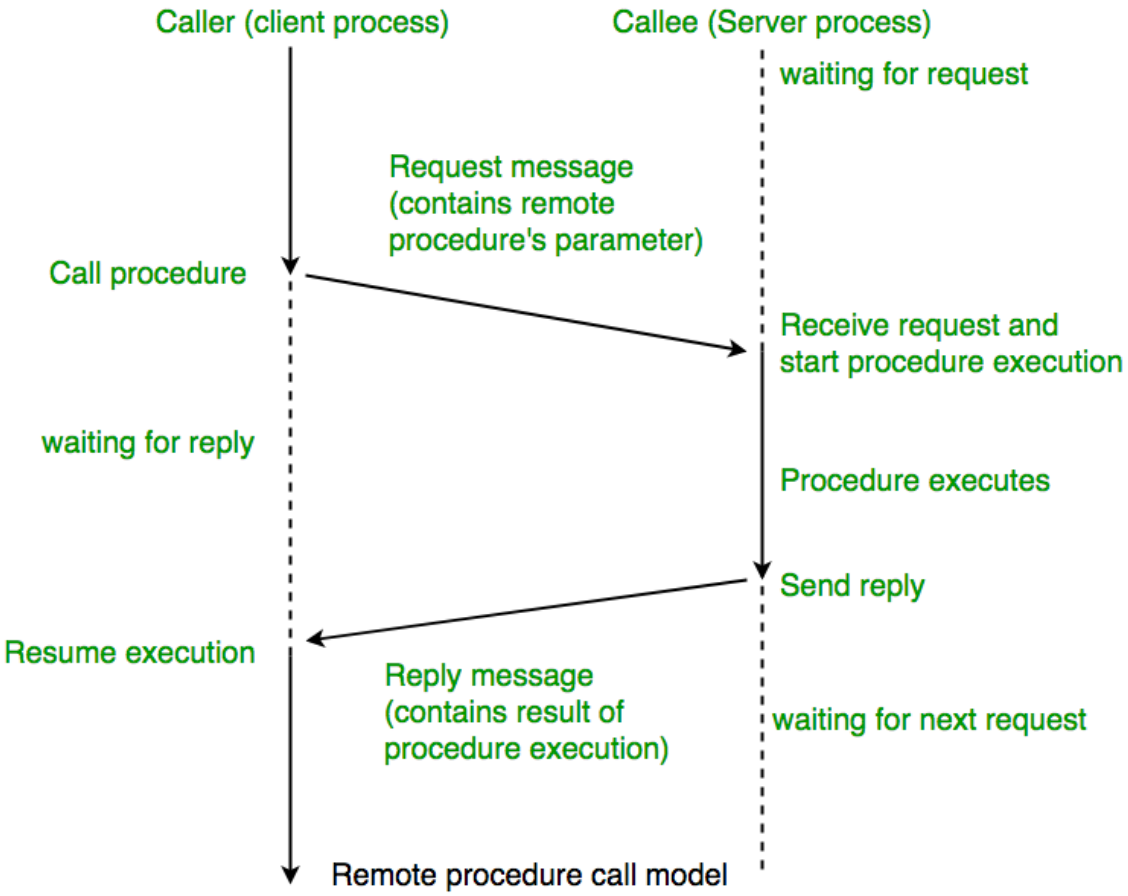
Experiment – 8

AIM: Implement RPC.
For implementing RPC use "rpcgen" to create client and server stubs.
The Remote Procedure should return the SUM, DIFFERENCE, MULTIPLE and DIVISION of two numbers to the process that has initiated the RPC

Theory:

Remote Procedure Call (RPC) is a powerful technique for constructing distributed, client-server based applications. It is based on extending the conventional local procedure calling so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them.

When making a Remote Procedure Call:

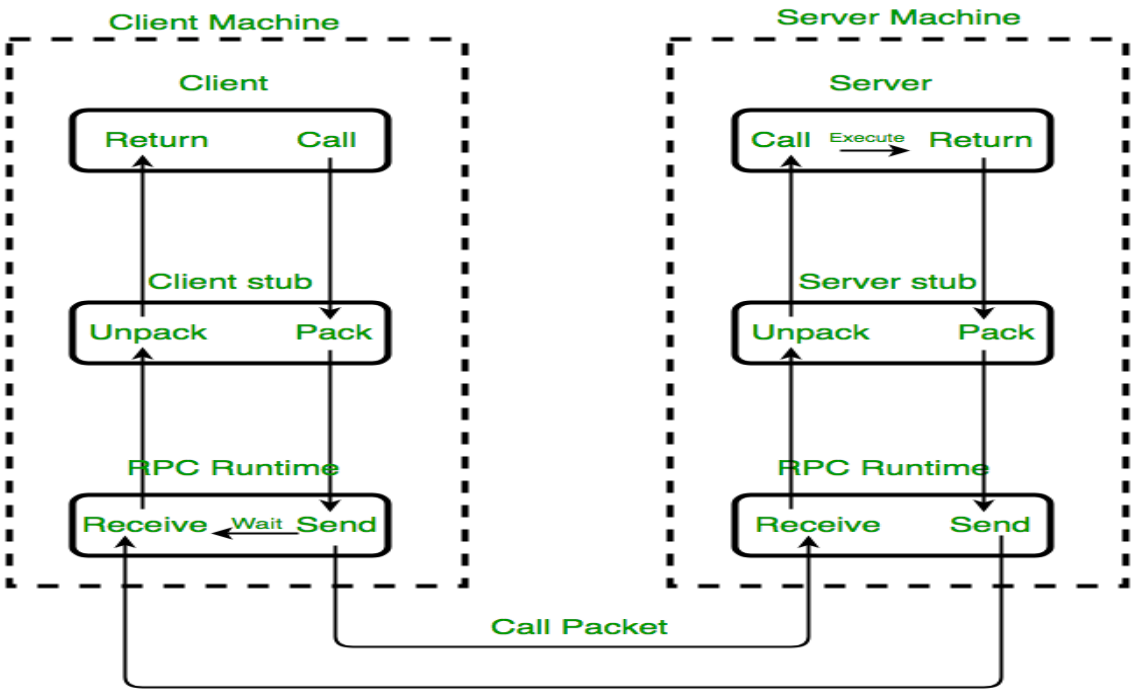


1. The calling environment is suspended, procedure parameters are transferred across the network to the environment where the procedure is to execute, and the procedure is executed there.

2. When the procedure finishes and produces its results, its results are transferred back to the calling environment, where execution resumes as if returning from a regular procedure call.

NOTE: RPC is especially well suited for client-server (e.g. query-response) interaction in which the flow of control alternates between the caller and callee. Conceptually, the client and server do not both execute at the same time. Instead, the thread of execution jumps from the caller to the callee and then back again.

Working of RPC:



Implementation of RPC mechanism

Code:

Server Process:

```
import sys
from xmlrpc.server import SimpleXMLRPCServer
def sub(a, b):
    return a - b
def add(a, b):
    return a + b
def mul(a, b):
    return a * b
def div(a, b):
    return a / b
def foo(self):
    print("foo")

server= SimpleXMLRPCServer(("localhost",6789))
server.register_function(sub,"sub")
server.register_function(add,"add")

server.register_function(mul,"mul")
server.register_function(div,"div")
server.serve_forever()
```

Client Process:

```
import sys
import xmlrpc.client
port=xmlrpc.client.ServerProxy("http://localhost:6789/")

print("calculator")
print("1. Addition")
print("2. Subtration")
```

```
print("3. Multiplication")
print("4. Divison")
while(True):
    choice=input(" Enter your operation with operation to be performed(like 1 2 3 (2+3=5)) :
")
    ch=choice.split()
    if int(ch[0])==1:
        x=port.add(int(ch[1]),int(ch[2]))
        print("Addition : ",x)

    elif int(ch[0])==2:
        x = port.sub(int(ch[1]),int(ch[2]))
        print("subtraction : ",x)

    elif int(ch[0])==3:
        x = port.mul(int(ch[1]),int(ch[2]))
        print("Multiplication : ",x)

    elif int(ch[0])==4:
        try:
            x = port.div(int(ch[1]),int(ch[2]))
            print("Division : ",x)
        except ZeroDivisionError:
            print("Zero Division not Possible")

    else:
        print("Operation doesn't exist.")
        sys.exit()
#result=proxy.add(num1,num2)
#print(result)
```

OUTPUT:

SERVER:

```
PS H:\pw dsa> python -u "c:\Users\Ankit\Downloads\SERVERRPC.py"
127.0.0.1 - - [16/Nov/2022 21:43:03] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2022 21:43:59] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2022 21:44:09] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2022 21:44:49] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2022 22:05:07] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2022 22:05:18] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2022 22:05:25] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2022 22:05:44] "POST / HTTP/1.1" 200 -
```

CLIENT:

```
PS H:\pw dsa> python -u "c:\Users\Ankit\Downloads\clientrpc.py"
calculator
1. Addition
2. Subtration
3. Multiplication
4. Divison
 Enter your operation with operation to be performed(like 1 2 3 (2+3=5)) :
1 4 2
Addition :  6
 Enter your operation with operation to be performed(like 1 2 3 (2+3=5)) :
2 4 2
subtraction :  2
 Enter your operation with operation to be performed(like 1 2 3 (2+3=5)) :
3 4 2
Multiplication :  8
 Enter your operation with operation to be performed(like 1 2 3 (2+3=5)) :
4 4 2
Division :  2.0
 Enter your operation with operation to be performed(like 1 2 3 (2+3=5)) :
```