CS-6210-A -Advanced Operating Systems

Project 3

Nidish Nair | GT ID# 903055730　　**Akshat Harit | GT ID# 903090915**

Implementation

Description

We have implemented the MultiRPC by modifying the source code of the XML-RPC library. We worked with version number 1.25.30, which is currently the super-stable version. We have modified the example server code to fork off n number of servers on successive port numbers beginning from 8080 on localhost. As for the client side, we followed a general design principle of creating wrapper functions to be called from our example code. These wrapper functions create multiple threads that contact the servers in order to simulate the parallel sending of RPC requests. The responses from each of these request threads are processed depending on the semantic. The exact implementation details are described in subsequent sections.

How to Compile

Compile instructions are same as that of xml-rpc. We have to first compile the xml-rpc library

```
./configure --prefix=installation_base_directory
make
make install
```

For the example files, we just have to include the requisite header files, and compile as usual

How to Run

For our three example files, the run details are

```
./xmlrpc_sample_add_server INITIAL_PORT_NUMBER NUMBER_OF_SERVERS
```

NUMBER_OF_SERVERS: How many servers to create
INITIAL_PORT :Initial port number of server to create

For example, “xmlrpc_sample_add_server 8080 5”, would start five servers on localhost with port numbers 8080, 8081, 8082, 8083 and 8084.

```
./xmlrpc_asynch_client INITIAL_PORT_NUMBER NUMBER_OF_CALLS
```

NUMBER_OF_CALLS: How many servers to connect. (For other symbols, see above)

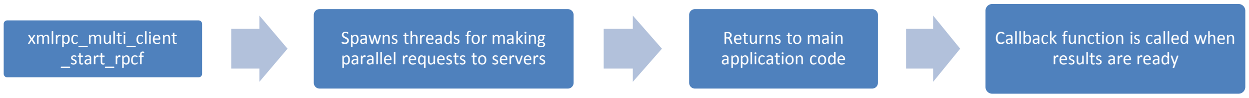
```
./xmlrpc_sample_add_client NUMBER_OF_CALLS INITIAL_PORT ARGUMENT1 ARGUMENT2
```

ARGUMENT1, ARGUMENT2 : Arguments for sample.add in the sample application((For other symbols, see above)

Asynchronous MultiRPC

Example file - xmlrpc_asynch_client.c

Asynchronicity is maintained when the calling function can move on to other tasks after making the initial call and be notified when the results are ready. The flow of function calls is as follows -


xmlrpc_multi_client _start_rpcf
Spawns threads for making parallel requests to servers
Returns to main application code
Callback function is called when results are ready

We implement our custom made function here, xmlrpc_multi_client_start_rpcf() to which we pass the number of servers to contact and the call semantics (any, majority, all), in addition to the usual parameter required for an RPC.

```
xmlrpc_multi_client_start_rpcf(&env, clientP, semantic, num_requests, methodName,
                    &handle_sample_add_response , NULL,
                    format", arg1, arg2);
```

The additions are semantic and num_requests.

This function spawns threads for making parallel requests to servers and once the requests have been made, the control flow reaches back to the application code in the example file. At this point any other computation can be made by the application, since the previous calls were async.

We've modified the function asynchComplete, which is where we've made the code changes for returning results as per the semantics. More details in the Semantics section.

Synchronous MultiRPC

Example file - xmlrpc_sample_add_client.c
This file gives an example of synchronous rpc call. The call made is
resultP=xmlrpc_client_call_multi(&env, clientP, type, number_of_calls , urls, methodName, "format", arg1, arg2)
The additional arguments are type, number_of_calls and urls. Type represents the semantic and accepts 'a', 'l' and 'm'. Number_of_calls is the number of server urls to call, while urls is an array of strings that presents the server urls.
The flow of function is as follows. When we call our function, we define an argument data struct and then create threads for each call. The threads call a wrapper function xmlrpc_client_call_2f_va_wrapper() that just deconstructs the argument struct and then calls xmlrpc_client_call_2f_va(). This is a standard library function for sync calls.
Then we try to join the threads as detailed in the following section.

Any, Majority and All Semantics

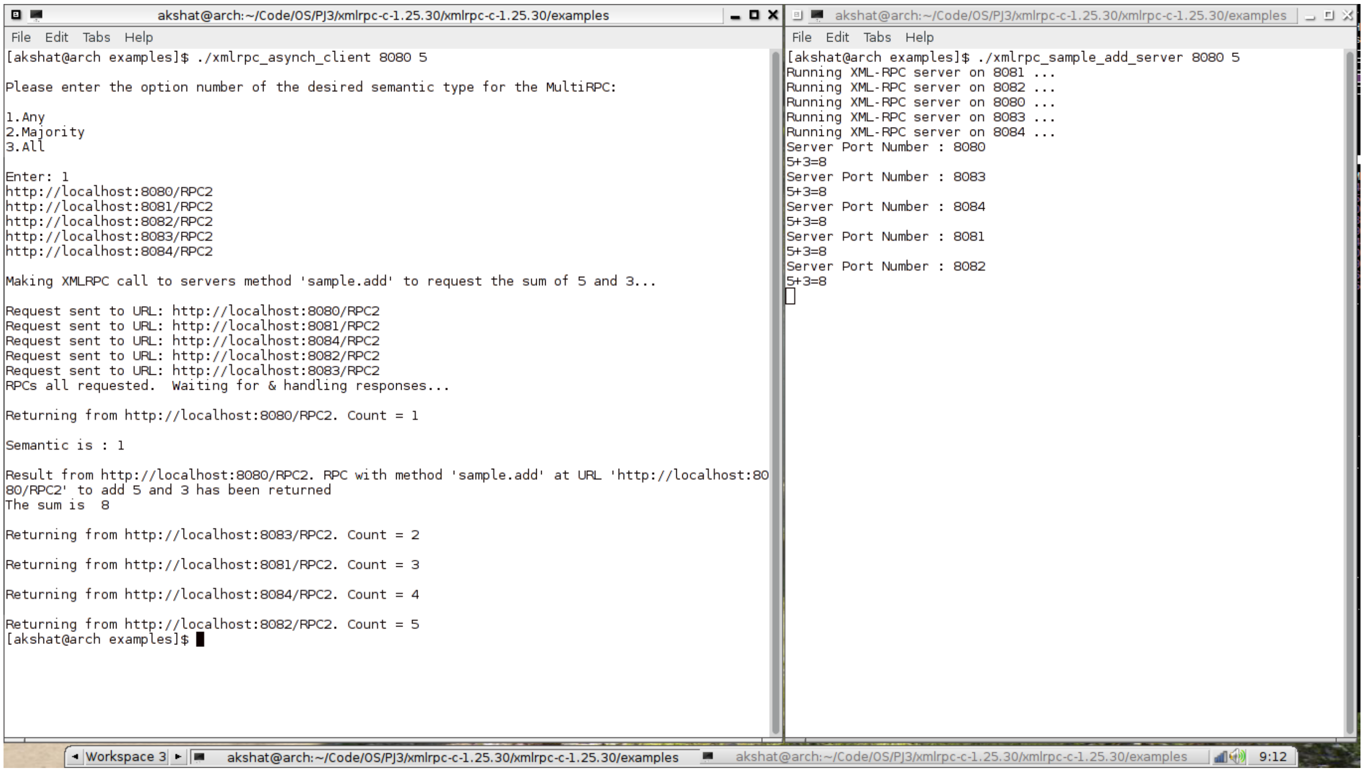
For synchronous calls, for any and majority semantics, the calling function after creating different threads for each server, tries to join them. A pthread can only be joined if it has completed. So for any semantic, the first pthread to successfully join results in returning the value of that thread. For majority semantic, a majority number of pthread are joined and then the result returned. The non-blocking function pthread_tryjoin_np is used in this case instead of pthread_join. This is done to ensure that we do not unnecessarily wait for a pthread and instead join the first one to complete. We continually iterate over the threads created, trying the non-blocking call. We return when sufficient number of calls have been returned.

For 'all' semantic, we join all the threads and then return.

For asynchronous calls, we maintain a global counter in asynch_complete function. Whenever the server returns a result, this function is invoked and within it, we increment the counter. Depending on the expected value of this counter, i.e. 1 in case of any, majority in case of majority and all requests in case of all semantic, we call the response handler function. So, if 5 RPC requests have been made, the application side response handler is called as soon as 3 responses have been received from the server.

Demonstration

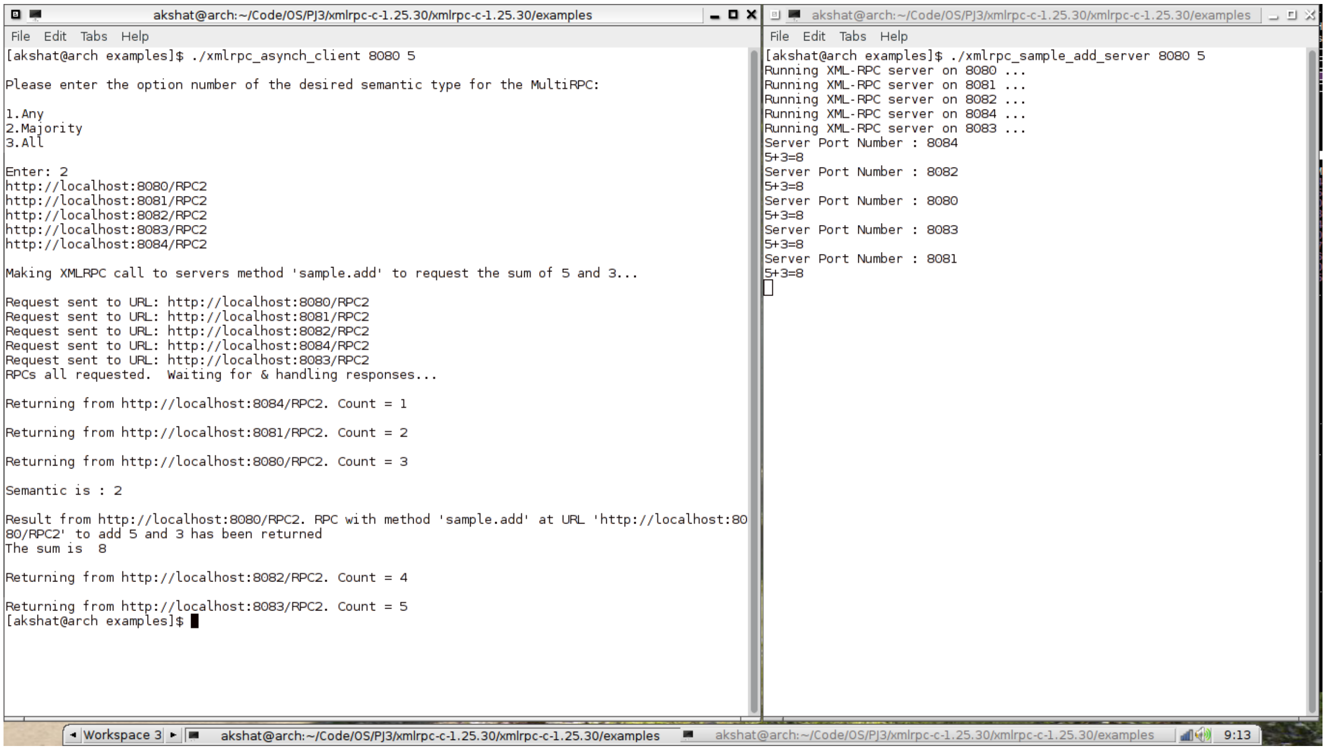
Async Any


akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
Workspace 3
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
9:12

```
[akshat@arch examples]$ ./xmlrpc_asynch_client 8080 5

Please enter the option number of the desired semantic type for the MultiRPC:

1.Any
2.Majority
3.All

Enter: 1
http://localhost:8080/RPC2
http://localhost:8081/RPC2
http://localhost:8082/RPC2
http://localhost:8083/RPC2
http://localhost:8084/RPC2

Making XMLRPC call to servers method 'sample.add' to request the sum of 5 and 3...

Request sent to URL: http://localhost:8080/RPC2
Request sent to URL: http://localhost:8081/RPC2
Request sent to URL: http://localhost:8084/RPC2
Request sent to URL: http://localhost:8082/RPC2
Request sent to URL: http://localhost:8083/RPC2
RPCs all requested.  Waiting for & handling responses...

Returning from http://localhost:8080/RPC2. Count = 1

Semantic is : 1

Result from http://localhost:8080/RPC2. RPC with method 'sample.add' at URL 'http://localhost:80
80/RPC2' to add 5 and 3 has been returned
The sum is  8

Returning from http://localhost:8083/RPC2. Count = 2

Returning from http://localhost:8081/RPC2. Count = 3

Returning from http://localhost:8084/RPC2. Count = 4

Returning from http://localhost:8082/RPC2. Count = 5
[akshat@arch examples]$ 
```

```
[akshat@arch examples]$ ./xmlrpc_sample_add_server 8080 5
Running XML-RPC server on 8081 ...
Running XML-RPC server on 8082 ...
Running XML-RPC server on 8080 ...
Running XML-RPC server on 8083 ...
Running XML-RPC server on 8084 ...
Server Port Number : 8080
5+3=8
Server Port Number : 8083
5+3=8
Server Port Number : 8084
5+3=8
Server Port Number : 8081
5+3=8
Server Port Number : 8082
5+3=8
```

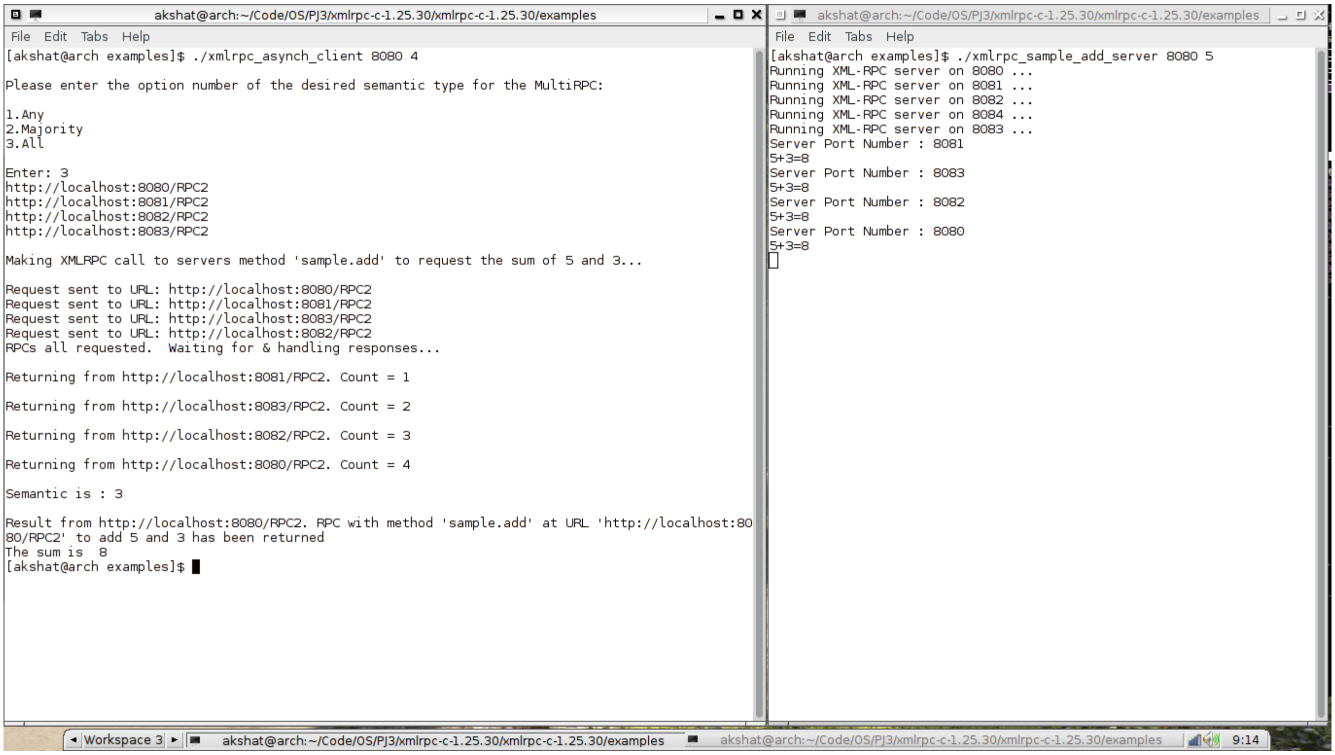
Async Majority


akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
Workspace 3
9:13

```
[akshat@arch examples]$ ./xmlrpc_asynch_client 8080 5

Please enter the option number of the desired semantic type for the MultiRPC:

1.Any
2.Majority
3.All

Enter: 2
http://localhost:8080/RPC2
http://localhost:8081/RPC2
http://localhost:8082/RPC2
http://localhost:8083/RPC2
http://localhost:8084/RPC2

Making XMLRPC call to servers method 'sample.add' to request the sum of 5 and 3...

Request sent to URL: http://localhost:8080/RPC2
Request sent to URL: http://localhost:8081/RPC2
Request sent to URL: http://localhost:8082/RPC2
Request sent to URL: http://localhost:8084/RPC2
Request sent to URL: http://localhost:8083/RPC2
RPCs all requested.  Waiting for & handling responses...

Returning from http://localhost:8084/RPC2. Count = 1

Returning from http://localhost:8081/RPC2. Count = 2

Returning from http://localhost:8080/RPC2. Count = 3

Semantic is : 2

Result from http://localhost:8080/RPC2. RPC with method 'sample.add' at URL 'http://localhost:80
80/RPC2' to add 5 and 3 has been returned
The sum is  8

Returning from http://localhost:8082/RPC2. Count = 4

Returning from http://localhost:8083/RPC2. Count = 5
[akshat@arch examples]$
```

```
[akshat@arch examples]$ ./xmlrpc_sample_add_server 8080 5
Running XML-RPC server on 8080 ...
Running XML-RPC server on 8081 ...
Running XML-RPC server on 8082 ...
Running XML-RPC server on 8084 ...
Running XML-RPC server on 8083 ...
Server Port Number : 8084
5+3=8
Server Port Number : 8082
5+3=8
Server Port Number : 8080
5+3=8
Server Port Number : 8083
5+3=8
Server Port Number : 8081
5+3=8
```

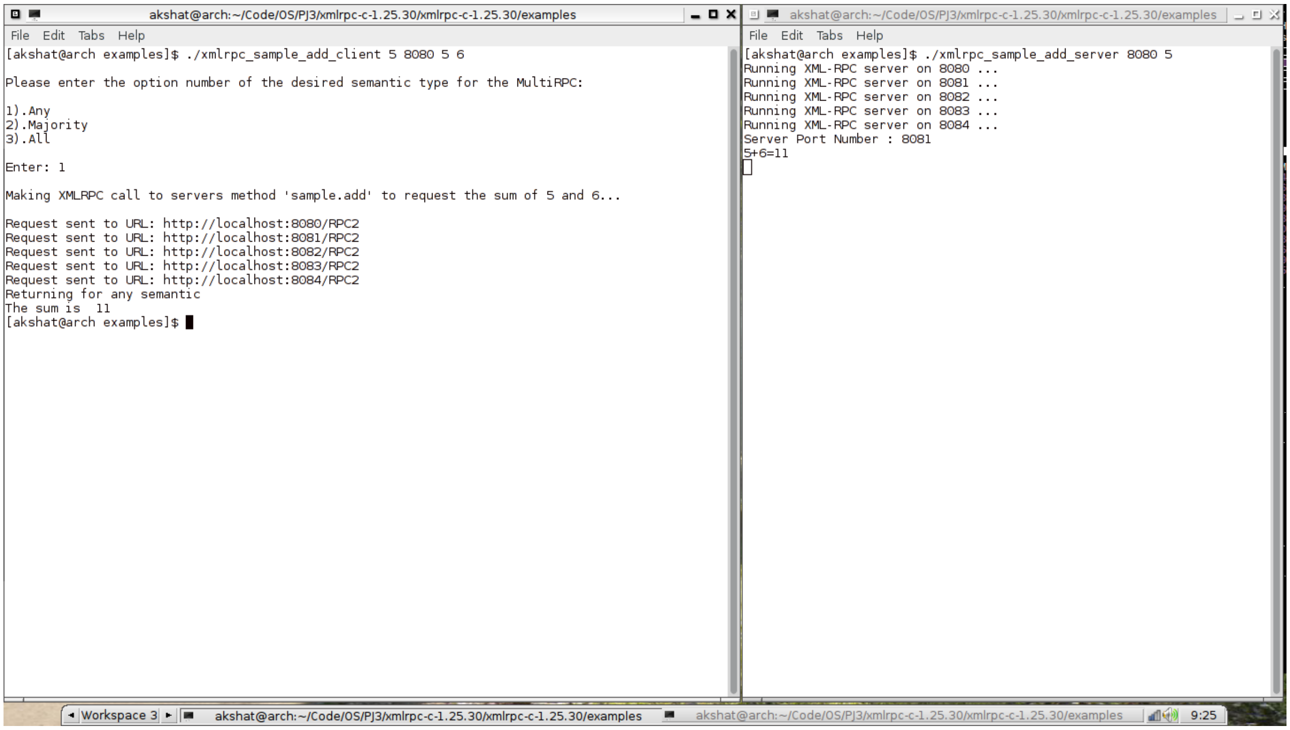
Async All


akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
Workspace 3
9:14

```
[akshat@arch examples]$ ./xmlrpc_asynch_client 8080 4

Please enter the option number of the desired semantic type for the MultiRPC:

1.Any
2.Majority
3.All

Enter: 3
http://localhost:8080/RPC2
http://localhost:8081/RPC2
http://localhost:8082/RPC2
http://localhost:8083/RPC2

Making XMLRPC call to servers method 'sample.add' to request the sum of 5 and 3...

Request sent to URL: http://localhost:8080/RPC2
Request sent to URL: http://localhost:8081/RPC2
Request sent to URL: http://localhost:8083/RPC2
Request sent to URL: http://localhost:8082/RPC2
RPCs all requested.  Waiting for & handling responses...

Returning from http://localhost:8081/RPC2. Count = 1

Returning from http://localhost:8083/RPC2. Count = 2

Returning from http://localhost:8082/RPC2. Count = 3

Returning from http://localhost:8080/RPC2. Count = 4

Semantic is : 3

Result from http://localhost:8080/RPC2. RPC with method 'sample.add' at URL 'http://localhost:80
80/RPC2' to add 5 and 3 has been returned
The sum is  8
[akshat@arch examples]$
```

```
[akshat@arch examples]$ ./xmlrpc_sample_add_server 8080 5
Running XML-RPC server on 8080 ...
Running XML-RPC server on 8081 ...
Running XML-RPC server on 8082 ...
Running XML-RPC server on 8084 ...
Running XML-RPC server on 8083 ...
Server Port Number : 8081
5+3=8
Server Port Number : 8083
5+3=8
Server Port Number : 8082
5+3=8
Server Port Number : 8080
5+3=8
```

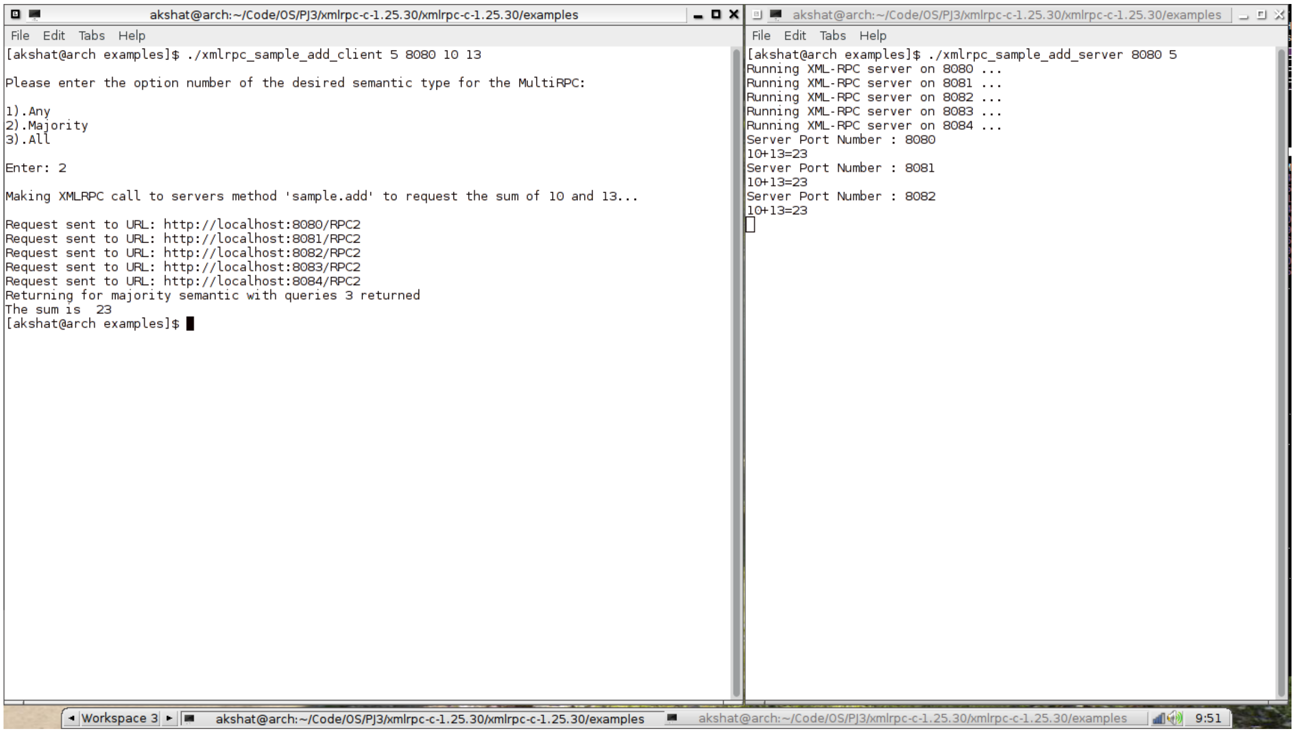
Sync Any


akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
Workspace 3
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
9:25

```
[akshat@arch examples]$ ./xmlrpc_sample_add_client 5 8080 5 6

Please enter the option number of the desired semantic type for the MultiRPC:

1).Any
2).Majority
3).All

Enter: 1

Making XMLRPC call to servers method 'sample.add' to request the sum of 5 and 6...

Request sent to URL: http://localhost:8080/RPC2
Request sent to URL: http://localhost:8081/RPC2
Request sent to URL: http://localhost:8082/RPC2
Request sent to URL: http://localhost:8083/RPC2
Request sent to URL: http://localhost:8084/RPC2
Returning for any semantic
The sum is  11
[akshat@arch examples]$
```

```
[akshat@arch examples]$ ./xmlrpc_sample_add_server 8080 5
Running XML-RPC server on 8080 ...
Running XML-RPC server on 8081 ...
Running XML-RPC server on 8082 ...
Running XML-RPC server on 8083 ...
Running XML-RPC server on 8084 ...
Server Port Number : 8081
5+6=11
```

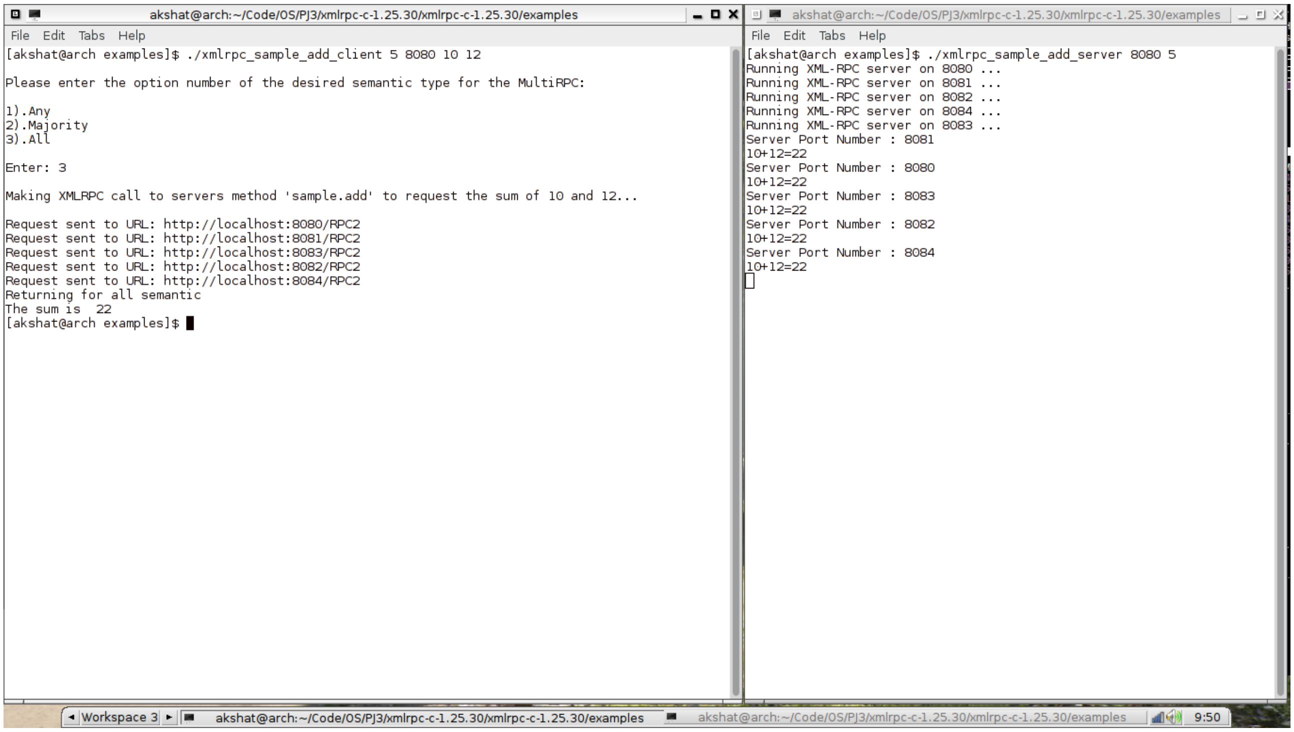
Sync Majority


akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
Workspace 3
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
9:51

```
[akshat@arch examples]$ ./xmlrpc_sample_add_client 5 8080 10 13

Please enter the option number of the desired semantic type for the MultiRPC:

1).Any
2).Majority
3).All

Enter: 2

Making XMLRPC call to servers method 'sample.add' to request the sum of 10 and 13...

Request sent to URL: http://localhost:8080/RPC2
Request sent to URL: http://localhost:8081/RPC2
Request sent to URL: http://localhost:8082/RPC2
Request sent to URL: http://localhost:8083/RPC2
Request sent to URL: http://localhost:8084/RPC2
Returning for majority semantic with queries 3 returned
The sum is  23
[akshat@arch examples]$
```

```
[akshat@arch examples]$ ./xmlrpc_sample_add_server 8080 5
Running XML-RPC server on 8080 ...
Running XML-RPC server on 8081 ...
Running XML-RPC server on 8082 ...
Running XML-RPC server on 8083 ...
Running XML-RPC server on 8084 ...
Server Port Number : 8080
10+13=23
Server Port Number : 8081
10+13=23
Server Port Number : 8082
10+13=23
```

Sync All


akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
File Edit Tabs Help
Workspace 3
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
akshat@arch:~/Code/OS/PJ3/xmlrpc-c-1.25.30/xmlrpc-c-1.25.30/examples
9:50

```
[akshat@arch examples]$ ./xmlrpc_sample_add_client 5 8080 10 12

Please enter the option number of the desired semantic type for the MultiRPC:

1).Any
2).Majority
3).All

Enter: 3

Making XMLRPC call to servers method 'sample.add' to request the sum of 10 and 12...

Request sent to URL: http://localhost:8080/RPC2
Request sent to URL: http://localhost:8081/RPC2
Request sent to URL: http://localhost:8083/RPC2
Request sent to URL: http://localhost:8082/RPC2
Request sent to URL: http://localhost:8084/RPC2
Returning for all semantic
The sum is  22
[akshat@arch examples]$
```

```
[akshat@arch examples]$ ./xmlrpc_sample_add_server 8080 5
Running XML-RPC server on 8080 ...
Running XML-RPC server on 8081 ...
Running XML-RPC server on 8082 ...
Running XML-RPC server on 8084 ...
Running XML-RPC server on 8083 ...
Server Port Number : 8081
10+12=22
Server Port Number : 8080
10+12=22
Server Port Number : 8083
10+12=22
Server Port Number : 8082
10+12=22
Server Port Number : 8084
10+12=22
```

Final Notes and Observations

It is observed that for the sample application of adding numbers across servers on the same machine (different ports), there is almost no difference in the execution times and so no useful analysis can be done in those terms. Hypothetically speaking, in a real-life situation, if there is a physical network between the client and the servers, there would have been more temporal data to analyze.

Also, currently, once a while, while making synchronous calls with **any** or **majority** to the server, this message pops up after all the RPC requests have been served on the server side - “Failed to read from Abyss connection. Error reading from channel”. This happens because in those 2 semantics, we tend to return the result to the application as soon as a certain number of responses (threads) are received. This leaves the remaining threads hanging, and that is causing this issue server side.

The obvious fix to this is to wait and close all threads even, but that violates the principle of returning as soon as “any” or “majority” of responses (threads) are received. In any case, as far as we’ve debugged, this server side message isn’t a serious issue and doesn’t interfere with the functioning of the RPC mechanism.