**Binder:**

The binder maintains database to store pairs of registered procedure and the list of server identifiers, which send registration request for the procedure. This database is implemented as a map such that:

- key is **ProcSignature** structure that contains procedure name, argTypes and the length of argTypes

- value is a list of **ProcLocation** structures that contains hostname, socket and port number of the

server

It also maintains list of servers (struct location) for round robin, and vector of active server sockets that will be closed if the server terminates.

Procedure overloading is handled implicitly by overloading equality and comparison operators for **ProcSignature** and **ProcLocation** structures. Hence, the same procedure with different argTypes are considered as function overloading and added to the database as a new entry.

If message type REGISTER is received, binder registers the procedure that server has sent. If the same server tries to register procedure with the same name and the same arguments in argTypes, then binder sends REGISTER\_FAILURE with warning code ERR\_RPC\_PROC\_RE\_REG to the server. Procedures with the same name and different arguments in argTypes are allowed be registered. In this case REGISTER\_SUCCESS with warning code ERR\_RPC\_SUCCESS will be sent to server. However, procedures with the same name and the same arguments in argTypes that differs only in their array lengths are considered as re-registration and REGISTER \_FAILURE with warning code ERR\_RPC\_PROC\_RE\_REG is sent to the server. The binder sends error code only if memory allocation in the heap fails during message reading. In this case REGISTER \_FAILURE with error code ERR\_BINDER\_OUT\_OF\_MEMORY is sent to the server and binder rejects registration.

If message type LOC\_REQUEST is received, binder searches for the requested procedure in the mapping table and if it is there, round robin algorithm will determine the next server for this procedure. Then binder sends LOC\_SUCCESS and server information to the client. Otherwise, binder sends LOC\_FAILURE with error code ERR\_RPC\_NO\_SERVER\_AVAIL to indicate that this is invalid procedure that has not been registered yet. If memory allocation in the heap fails during message reading, LOC\_FAILURE with error code ERR\_BINDER\_OUT\_OF\_MEMORY is sent to the client.

Binder adds a server to the list of servers for round robin only if there is more than one server that registered the same procedure. This is because there is no need for round robin algorithm to choose the next server for the procedure, which is only registered by single server. In this case that server is always chosen. Otherwise, round robin algorithm finds the next server and moves it to the end of the list.

If one of the active servers is terminated or disconnected, it will be removed from mapping table of procedure and servers, list of server identifiers and vector of active server sockets.

If message type TERMINATE is received, binder sends terminate message to all servers in the vector of active server sockets. Then binder closes its socket and terminates. Also, if the binder suddenly disconnects, all servers that are connected to the binder will be terminated.

**Error codes:**

|  |  |  |
| --- | --- | --- |
| Return code | Value | Description |
| ERR\_RPC\_PROC\_RE\_REG | 1 | Warning: procedure has already been registered. |
| ERR\_RPC\_SUCCESS | 0 | Success |
| ERR\_RPC\_OUT\_OF\_MEMORY | -1 | Error: no memory available on heap of the rpc |
| ERR\_RPC\_PROC\_EXEC\_FAILED | -2 | Error: procedure returns an error |
| ERR\_RPC\_EXEC\_BEFORE\_REG | -3 | Error: procedure is executed before registration |
| ERR\_RPC\_UNREGISTERED\_PROC | -4 | Error: invalid procedure |
| ERR\_RPC\_NO\_SERVER\_AVAIL | -5 | Error: invalid procedure request (no available server) |
| ERR\_RPC\_UNEXPECTED\_MSG\_TYPE | -6 | Error: invalid message type |
| ERR\_RPC\_UNKNOWN\_TERMINATION\_SRC | -8 | Error: |
| ERR\_RPC\_SOCKS\_INACTIVE | -9 | Error: no active sockets |
| ERR\_RPC\_SOCKET\_FAILED | -10 | Error: invalid socket |
| ERR\_RPC\_ENV\_ADDR\_NULL | -11 | Error: invalid binder hostname |
| ERR\_RPC\_ENV\_PORT\_NULL | -12 | Error: invalid binder port |
| ERR\_RPC\_HOSTENT\_NULL | -13 | Error: invalid binder hostent |
| ERR\_RPC\_THREAD\_NOT\_CREATED | -14 | Error: pthread\_create() is failed |
| ERR\_BINDER\_OUT\_OF\_MEMORY | -15 | Error: no memory available on heap of the binder |
| ERR\_RPC\_BINDER\_SOCK\_CLOSED | -17 | Error: binder socket is closed |
| ERR\_RPC\_BINDER\_SOCK\_FAILED | -18 | Error: invalid binder socket |
| ERR\_RPC\_SERVER\_SOCK\_FAILED | -19 | Error: invalid server socket |
| ERR\_BINDER\_TERMINATE\_SIG | -50 | Error: terminate message is received, stop binder |
| ERR\_RPC\_INCOMPLETE\_MSG | -99 | Error: message size is not properly aligned |
| ERR\_RPC\_ARR\_LEN\_NOT\_ENOUGH | -100 | Error: |

**Unimplemented functionalities:**

Bonus functionality: rpcCacheCall() for the client. However, the binder handles cache requests.