1. **RPC Interface Definition (add.x):**

   This file defines the interface for the RPC service. It specifies the ADDER program, which contains one version (ADDVER). Within this version, there is one procedure ADD, which takes two integers as arguments and returns an integer (the sum of the two integers).

   First, we define the RPC interface in a file named add.x:

   ```
   // add.x

   program ADDPROG {
      version ADDVER {
         int ADD(int, int) = 1;
      } = 1;
   } = 0x31230000;
   ```

2. **Generating Stubs:**

   The rpcgen tool is used to generate client and server stubs from the interface definition file (add.x). The generated files (add_clnt.c, add_svc.c, add.h, add_xdr.c) contain the necessary code to handle communication between the client and server.

   Use the RPCGEN tool to generate client and server stubs:

   ```
   rpcgen add.x
   ```

   This command generates add_clnt.c, add_svc.c, add.h, and add_xdr.c.

3. **Server Implementation (add_server.c):**

- The server implements the add_1_svc function, which is called when the ADD procedure is invoked remotely. This function takes two integers as arguments (arg1 and arg2) and returns a pointer to an integer.
- Inside add_1_svc, the sum of arg1 and arg2 is computed and stored in a static variable result, which allows the result to persist across function calls.

The main function registers the ADD procedure with the RPC server (registerrpc) and starts the RPC server (svc_run) to handle incoming RPC requests

3. Implement the Server:

Create a file named add_server.c and implement the server logic:

```
#include <stdio.h>
#include "add.h"

int * add_1_svc(int *arg, struct svc_req *rqstp) {
   static int result;
   result = arg[0] + arg[1];
   return &result;
}

int main() {
   register SVCXPRT *transp;
```

```
    pmap_unset(ADDPROGRAM, ADDVERSION);
    transp = svcudp_create(RPC_ANYSOCK);
    if (transp == NULL) {
        fprintf(stderr, "cannot create udp service.\n");
        exit(1);
    }
    if (!svc_register(transp, ADDPROGRAM,
ADDVERSION, add_1, IPPROTO_UDP)) {
        fprintf(stderr, "unable to register (ADDPROGRAM,
ADDVERSION, udp).\n");
        exit(1);
    }
    svc_run();
    fprintf(stderr, "svc_run returned\n");
    exit(1);
    /* NOTREACHED */
}
```

4. **Client Implementation (add_client.c):**
    - The client program takes three command-line arguments: the hostname of the server, and two integers to be added.
    - It creates an RPC client handle (cl) using clnt_create, specifying the hostname of the server, the program number (ADDPROG), the version number (ADDVER), and the transport protocol ("udp" in this case).

- The client invokes the ADD procedure (add_1) remotely by passing the two integers as arguments. If successful, it receives a pointer to the result.
- Finally, the client prints the result obtained from the server and destroys the RPC client handle.

```
#include <stdio.h>
#include "add.h"
int main(int argc, char *argv[]) {
    CLIENT *cl;
    int *result;
    int args[2];

    if (argc != 4) {
        printf("Usage: %s hostname num1 num2\n", argv[0]);
        return 1;
    }
    cl = clnt_create(argv[1], ADDPROGRAM,
ADDVERSION, "udp");
    if (cl == NULL) {
        clnt_pcreateerror(argv[1]);
        return 1;
    }
    args[0] = atoi(argv[2]);
    args[1] = atoi(argv[3]);
    result = add_1(args, cl);
```

```
    if (result == NULL) {
        clnt_perror(cl, argv[1]);
        return 1;
    }
    printf("Result: %d\n", *result);
    clnt_destroy(cl);
    return 0;
}
```

5. **Compilation and Execution:**
    - The server and client programs are compiled using gcc, linking necessary libraries (-lnsl for socket programming).
    - The server program (add_server) is executed first to start the RPC server.
    - The client program (add_client) is then executed, providing the server's hostname and two integers to be added as command-line arguments.

    gcc -o server server.c add_svc.c add_xdr.c -lnsl

    gcc -o client client.c add_clnt.c add_xdr.c -lnsl

    Run the server:

    ./server

    Run the client

    ./client localhost 5 7

    Result: 12

The add.h file generated by rpcgen contains the declarations of the remote procedure call (RPC) function and the related data structures.

It serves as the interface between the client and server components of the RPC system.