Distributed Systems Report

Practical Work 2: RPC File Transfer

Nguyen Minh Cuong - 23BI14467

December 4, 2025

1 Introduction

The objective of this laboratory work is to upgrade the previous TCP file transfer system to use **Remote Procedure Calls (RPC)**. We utilized **ONC RPC (Sun RPC)** to abstract the networking layer, allowing the client to invoke a function on the server as if it were a local call. The system was implemented in C on the Kali Linux environment.

2 System Design

2.1 Protocol Design (IDL)

Unlike Lab 1 (Socket), we defined the protocol using an Interface Description Language (IDL) file named `transfer.x`.

- **Data Structure:** We defined a struct `file_data` containing:
  - `filename`: A string (max 256 chars) to identify the file.
  - `content`: An `opaque` type (variable-length bytes) to hold binary data.
  - `bytes_sent`: Integer indicating the file size.
- **Procedure:** The function `UPLOAD_FILE` takes `file_data` as input and returns an integer (1 for success).

2.2 System Organization

The architecture relies on the Stub/Skeleton model generated by `rpcgen`.

```
[Client Code] -> [Client Stub] -> (Network/XDR) -> [Server Skeleton] -> [Server Code]
```

3 Implementation

3.1 RPC Definition (transfer.x)

```
struct file_data {
    string filename<256>;
    opaque content<>;
    int bytes_sent;
};

program TRANSFER_PROG {
    version TRANSFER_VERS {
        int UPLOAD_FILE(file_data) = 1;
    } = 1;
} = 0x31230000;
```

Listing 1: The IDL file content

3.2 Client Logic (transfer_client.c)

The client reads the file into a buffer and calls the remote function.

```
// 1. Read file to buffer
file_buffer = (char *)malloc(file_size);
fread(file_buffer, 1, file_size, fp);

// 2. Pack data into RPC struct
args.filename = filename;
args.content.content_val = file_buffer;
args.content.content_len = file_size;

// 3. Call Remote Procedure
result = upload_file_1(&args, clnt);
```

Listing 2: Client packing data

3.3 Server Logic (transfer_server.c)

The server receives the data structure and writes it to the disk.

```
int * upload_file_1_svc(file_data *argp, struct svc_req *rqstp) {
    static int result;
    char new_name[300];

    sprintf(new_name, "rpc_recv_%s", argp->filename);

    FILE *fp = fopen(new_name, "wb");
    fwrite(argp->content.content_val, 1, argp->content.content_len, fp);
    fclose(fp);

    result = 1;
    return &result;
}
```

Listing 3: Server writing file

4 Results

The system successfully transferred a text file named `test_rpc.txt`. The server received it as `rpc_recv_test_rpc.txt`.

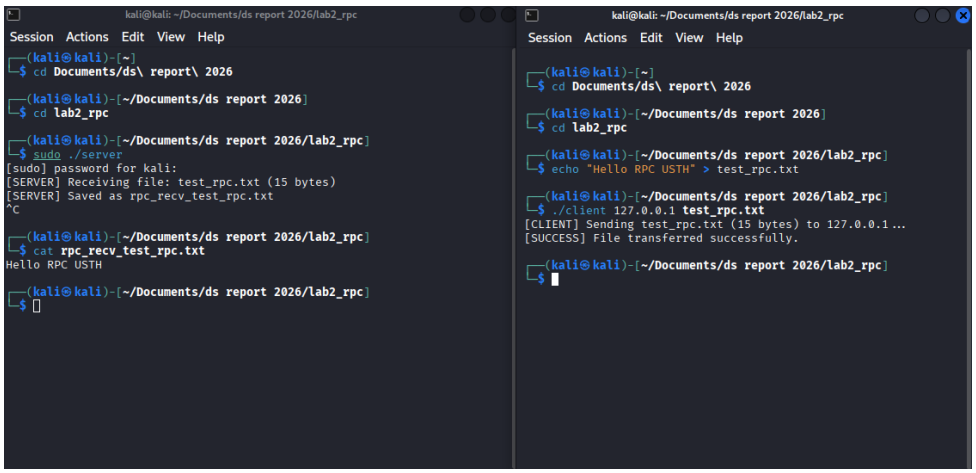

Figure 1: Screenshot of successful transfer on Kali Linux