Practical Work 2 - RPC File Transfering

Student name: Do Nhat Thanh
Student ID: BI12-416

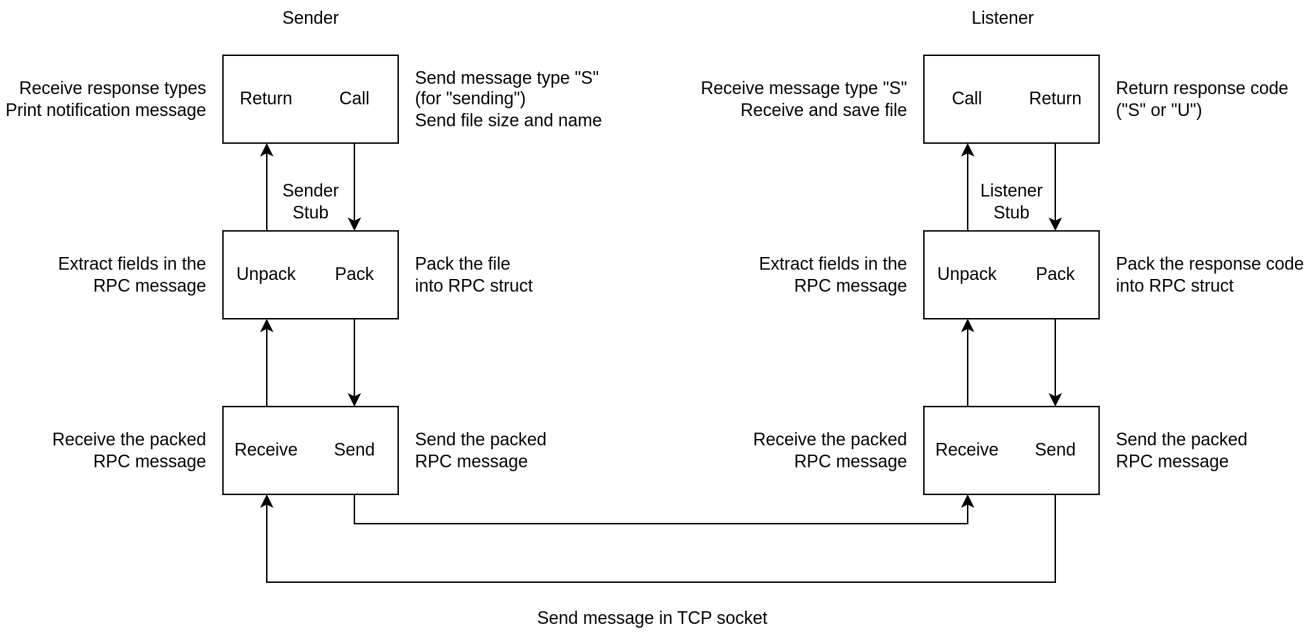
RPC Service Design

I define the RPC message consists of 3 parts:

- Type: a single message code
- Size: a number represents the size of the upcomming file to send
- File name: a string represents the file name

```c
typedef struct {
        char type; // 'M' for metadata
        size_t size; // Size of the file
        char filename[256]; // Name of the file
} RPCMessage;
```

File sending system using RPC



Implementing in C code

1. Sender

Defining RPC message

```
typedef struct {
    char type; // 'M' for metadata
    size_t size; // Size of the file
    char filename[256]; // Name of the file
} RPCMessage;
```

Preparing file metadata to send before sending the file content

```
RPCMessage metadata;
metadata.type = 'M';
strncpy(metadata.filename, basename(filename), sizeof(metadata.filename) -
1);
metadata.filename[sizeof(metadata.filename) - 1] = '\0';

struct stat st;
if (stat(filename, &st) == 0) {
        metadata.size = st.st_size;
} else {
        perror("Failed to get file size");
        close(sock);
        return 1;
}

send(sock, &metadata, sizeof(metadata), 0);
```

Sending the file content

```
FILE *file = fopen(filename, "rb");
if (file == NULL) {
        perror("Can't open file");
        close(sock);
        return 1;
}

char buffer[BUFFER_SIZE] = {0};
printf("Sending file: %s to %s:%d\n", filename, hostAddress, port);

while (!feof(file)) {
        int bytes_read = fread(buffer, 1, BUFFER_SIZE, file);
        if (bytes_read > 0) {
                send(sock, buffer, bytes_read, 0);
```

```
        }
}
```

Waiting for response code

```
RPCMessage response;
recv(sock, &response, sizeof(response), 0); // Wait for the confirmation
message

if (response.type == 'S') {
        printf("File has been successfully received by the server.\n");
} else {
        printf("An error occurred while transferring the file.\n");
}
```

2. Listener

Listening for the RPC message to prepare for receiving file

```
RPCMessage metadata;
recv(new_socket, &metadata, sizeof(metadata), 0);
```

Parsing the received RPC message, starting to receive file

```
if (metadata.type == 'M') {
        printf("Receiving file: %s, size: %lu\n", metadata.filename,
metadata.size);

        char filePath[512];
        snprintf(filePath, sizeof(filePath), "%s/%s", outputFolder,
metadata.filename);

        FILE *file = fopen(filePath, "wb");
        if (file == NULL) {
                perror("Failed to open file for writing");
                exit(EXIT_FAILURE);
        }

        // Receive file data
        char buffer[BUFFER_SIZE];
        size_t totalBytes = 0;
        while (totalBytes < metadata.size) {
                ssize_t bytesReceived = recv(new_socket, buffer,
```

```
BUFFER_SIZE, 0);
                if (bytesReceived <= 0) {
                        // Handle errors or connection closed by client
                        break;
                }
                fwrite(buffer, 1, bytesReceived, file);
                totalBytes += bytesReceived;
        }

        ...
        ...
```

Sending response code to sender

```
        RPCMessage response;
        response.type = 'S'; // Confirmation message type "S" for Success
        send(new_socket, &response, sizeof(response), 0);


        printf("File %s has been received and saved successfully.\n",
metadata.filename);

        fclose(file);
} else {
        RPCMessage response;
        response.type = 'U'; // Confirmation message type "U" for Unsuccess
        send(new_socket, &response, sizeof(response), 0);
        printf("Unexpected message type received.\n");
}
```