Practical Work 2: RPC File Transfer

Le Ba Hai Long 22bi13261

December 12, 2024

1 Introduction

This practical work upgrades the TCP-based file transfer system to use Remote Procedure Call (RPC). The simplified implementation uses gRPC for file transfer between a client and server.

2 System Design

The system is designed as follows:

1. The client reads the file and sends the entire file (name and content) to the server in one request.
2. The server receives the file and saves it locally.
3. The server responds with a success message.

3 Implementation

The implementation is done in Python using gRPC.

3.1 Server Code

Listing 1: Server Code

```
import grpc
from concurrent import futures
import file_transfer_pb2
import file_transfer_pb2_grpc
```

```
class FileTransferService(file_transfer_pb2_grpc.FileTransferServicer)
    def SendFile(self, request, context):
        with open(request.file_name, "wb") as f:
            f.write(request.content)
        print(f"File received and saved")
        return file_transfer_pb2.TransferStatus(success=True, message=

def serve():
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=1))
    file_transfer_pb2_grpc.add_FileTransferServicer_to_server(FileTran
    server.add_insecure_port("[::]:50051")
    server.start()
    server.wait_for_termination()

if __name__ == "__main__":
    serve()
```

3.2 Client Code

Listing 2: Client Code

```
import grpc
import file_transfer_pb2
import file_transfer_pb2_grpc

def send_file(name):
    with open(name, "rb") as f:
        content = f.read()

    channel = grpc.insecure_channel("localhost:50051")
    stub = file_transfer_pb2_grpc.FileTransferStub(channel)
    request = file_transfer_pb2.FileRequest(name=name, content=content
    response = stub.SendFile(request)

if __name__ == "__main__":
    send_file("prac2.txt")
```