Practical Work 3: MPI File Transfer

Group ID: 16

December 16, 2025

1 Introduction

The objective of this practical work is to upgrade a standard file transfer system into a distributed version using the Message Passing Interface (MPI). The system allows a file to be transferred from a sender process to a receiver process running in parallel.

2 Implementation Choice

We selected Python and the mpi4py library for this project. This choice allows us to leverage Python's simplicity for file I/O operations while utilizing the robust MPI standard bindings provided by mpi4py for process communication. It supports sending both serialized Python objects (for metadata) and binary buffers (for file content).

3 System Design

3.1 Architecture

The system adopts a master-slave communication model where Rank 0 acts as the Sender and Rank 1 acts as the Receiver.

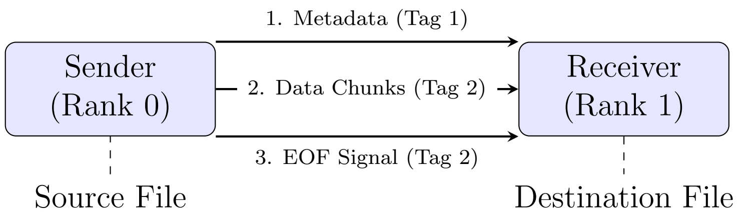

Figure 1: Data Flow Diagram

4 Implementation Details

4.1 Sender Logic

The sender reads the file and transmits it in chunks.

```python
def sender(self, filename):
    size = os.path.getsize(filename)
    self.comm.send({'filename': filename, 'filesize': size}, dest=1, tag
=1)
```

```

    with open(filename, 'rb') as f:
        while True:
            chunk = f.read(self.buffer_size)
            if not chunk: break
            self.comm.send(chunk, dest=1, tag=2)

    self.comm.send(None, dest=1, tag=2)
```

4.2 Receiver Logic

The receiver listens for metadata and then loops to receive content chunks.

```
def receiver(self):
    meta = self.comm.recv(source=0, tag=1)
    new_name = "received_" + meta['filename']

    with open(new_name, 'wb') as f:
        while True:
            chunk = self.comm.recv(source=0, tag=2)
            if chunk is None: break
            f.write(chunk)
```

5 Roles and Responsibilities

Member	**Role**	**Tasks**
Member 1	Architect	Designed the communication protocol and data flow.
Member 2	Developer	Implemented the file I/O and MPI logic.
Member 3	Analyst	Verified the file integrity and wrote the report.

Table 1: Group Task Allocation