Distributed Systems

Assignment-2 (Totally ordered Broadcast)

1.1 Introduction

The assignment focuses on the implementation and proof of a totally ordered broadcast protocol which ensures that the messages that are shared across the network are processed in order on all the processes. The implementation has a block which represents a process which internally has a middleware and an application. Each middleware and application has ports from where they can communicate with each other and the network.

The application has a port on which it listens for the incoming messages from the middleware for processing. Also, there is a port on which the application emits the messages that the application generates.The middleware has a total of two ports. One port is where it listens to incoming requests from the application and broadcasts that message over the network. Another port is where it listens for incoming messages from the network here the messages are divided in two parts one is a normal message and another is the acknowledgement where acknowledgement messages are just for the messages
in the queue. After receiving the message it is added to the queue and sorted according to the lamport timestamps. Now the clock is adjusted and another port is where the message is transmitted to the application for processing.

The queue thread is where the processing happens here it is checked that the message that is at the top of queue has received all acknowledgements then it is processed
else we deliver the ack messages over the network indicating that the message is at the top of the queue for the current process.

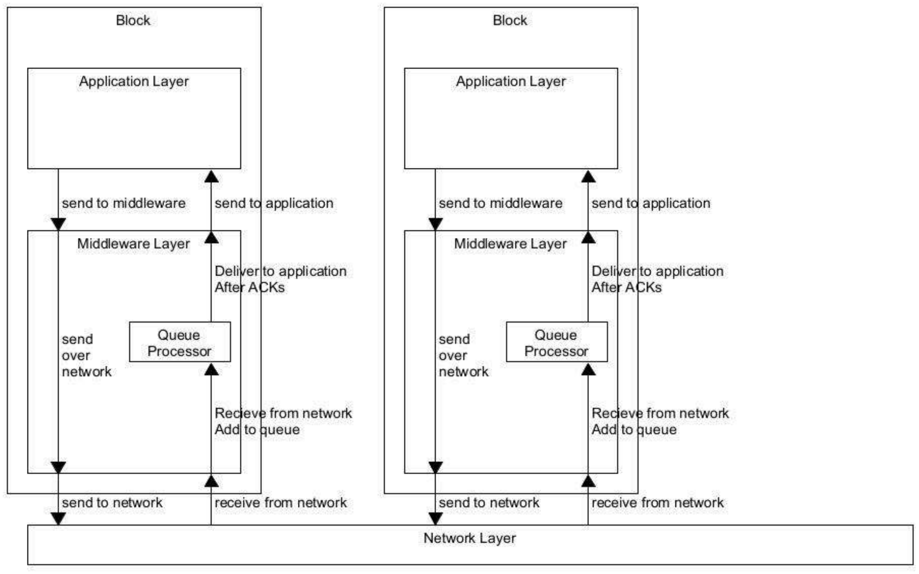
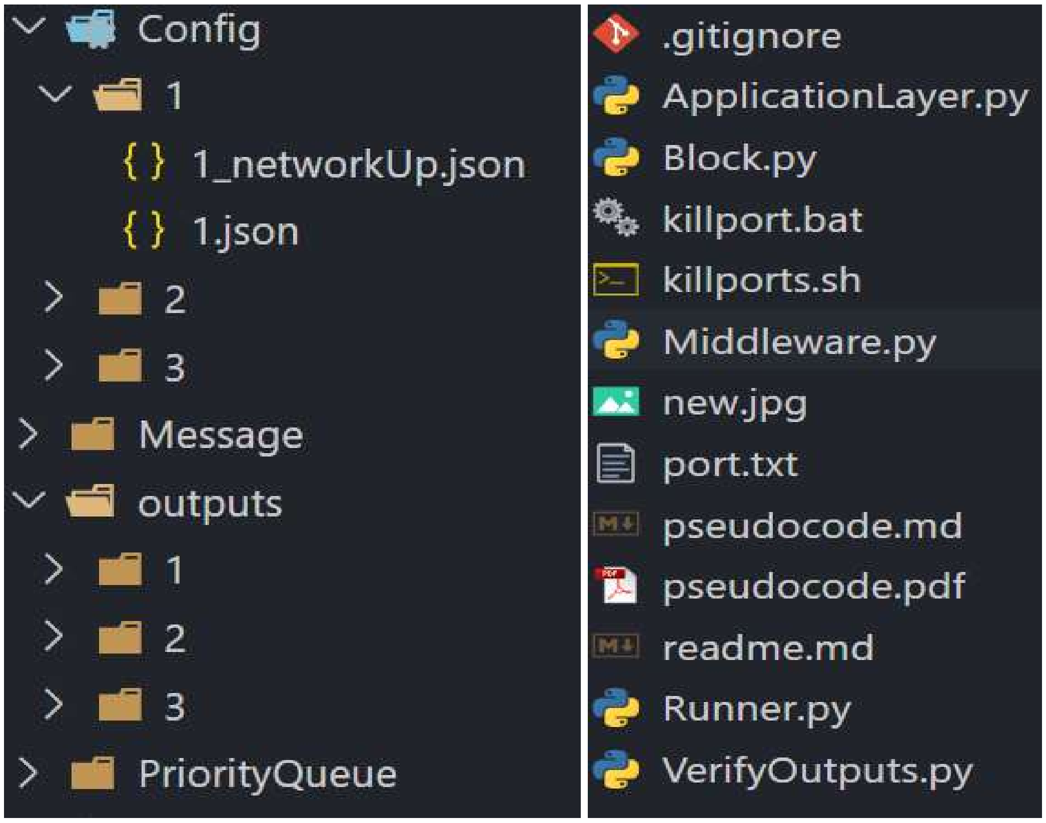


Fig-1: The above diagram represents the architecture of the implementation.

1.2 Implementation Details

The totally ordered broadcast is implemented by spawning blocks that would internally spawn a middleware block and an application block. Each middleware and application can communicate with each other using ports. The middleware is connected to the network layer in order to send the broadcast messages. Also, each message is a broadcast message be it a normal message or an acknowledgement message.

Below given is the file structure of the implementation. Where the Runner.py File is the main entrypoint file for the testing to test we use the command “py Runner.py <TESTCASE>”. Before using the Runner file run the file killports.bat/sh based on the environment you are running the program on. This would kill processes that were running on that port. The config folder contains all the testcases where each test case has a port configuration file indicating the number of test cases. Each test case under the config file has two files names <TESTCASE>.json which has the number of clients and the port configuration and another file <TESTCASE>_networkUp.json which has all the network receive ports for all process which is used to broadcast message. The outputs of the clients are stored inside the outputs folder which internally has folders named <TESTCASE> and inside that folder we have text files with <CLIENTID>.txt which has the order in which each client processes the messages. To verify the outputs we have the file named “VerifyOutputs.py <TESTCASE>” where it would evaluate the output files of the particular test case and output passed or failed based on the order the messages are processed.



```
Config
  1
    1_networkUp.json
    1.json
  2
  3
Message
outputs
  1
  2
  3
PriorityQueue
```

```
.gitignore
ApplicationLayer.py
Block.py
killport.bat
killports.sh
Middleware.py
new.jpg
port.txt
pseudocode.md
pseudocode.pdf
readme.md
Runner.py
VerifyOutputs.py
```

Fig-2: Overview of the file structure.

1.3 Testing Instructions

1. To check if the ports required for the testing are free.

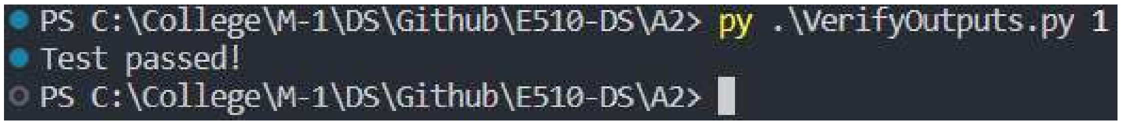
a. If on a Linux system run the file _killports.sh_ and provide the ports as a command line argument in a string example usage: ./killports.sh "3000 3001".

b. If on a Windows system run the file same as Linux but use the file _killport.bat_.

2. After we kill the ports remove the files inside the outputs folder or ensure the files are empty.

3. Now run the file Runner.py <TESTCASE> in the main folder. Run it and the processing order can be observed in the output folder under outputs/<TESTCASE> folder where the file name is x.txt where x represents a process say 1.txt would mean the order in which the block 1 processed the data. After the processes have run for a significant amount of time kill it and follow step-4.

4. To verify the outputs run the file VerifyOutputs.py which would internally read the output files and compare the processing order for each file and output passed or failed. Usage is py VerifyOutputs.py <TESTCASE> this would verify a particular <TESTCASE>.



```
PS C:\College\M-1\DS\Github\E510-DS\A2> py .\VerifyOutputs.py 1
Test passed!
PS C:\College\M-1\DS\Github\E510-DS\A2>
```

Fig-3: Output from the VerifyOutputs.py file.

1.4 Test cases

The test case files are under the Config folder and each folder has a <TESTCASE>.json file having the below given details for each client

1. Application
    a. MiddlewareUp
        i. Port
        ii. Host
    b. MiddlewareDown
        i. Port
        ii. Host
2. Middleware
    a. ApplicationUp
        i. Port
        ii. Host

    b. ApplicationDown
        i. Port
        ii. Host
    c. NetworkUp
        i. Port
        ii. Host
    d. NetworkDown
        i. Port
        ii. Host

1.4.1 Test case 1

Input:

```
{
  "1": {
    "Application": {
      "MiddlewareUp": {
        "port": 3000,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3001,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3000,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3001,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5000,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5001,
        "host": "localhost"
      }
    }
  },
  "2": {
    "Application": {
      "MiddlewareUp": {
        "port": 3002,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3003,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3002,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3003,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5002,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5003,
        "host": "localhost"
      }
    }
  },
  "3": {
    "Application": {
      "MiddlewareUp": {
        "port": 3004,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3005,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3004,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3005,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5004,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5005,
        "host": "localhost"
      }
    }
  }
}
```

Outputs:

1.txt

```
1.) Application processesA1 On block ID:1
2.) Application processesA2 On block ID:1
3.) Application processesA2 On block ID:1
4.) Application processesA3 On block ID:1
5.) Application processesA1 On block ID:1
6.) Application processesA2 On block ID:1
7.) Application processesA3 On block ID:1
8.) Application processesA1 On block ID:1
9.) Application processesA2 On block ID:1
```

2.txt

```
1.) Application processesA1 On block ID:2
2.) Application processesA2 On block ID:2
3.) Application processesA2 On block ID:2
4.) Application processesA3 On block ID:2
5.) Application processesA1 On block ID:2
6.) Application processesA2 On block ID:2
7.) Application processesA3 On block ID:2
8.) Application processesA1 On block ID:2
9.) Application processesA2 On block ID:2
```

3.txt

```
1.) Application processesA1 On block ID:3
2.) Application processesA2 On block ID:3
3.) Application processesA2 On block ID:3
4.) Application processesA3 On block ID:3
5.) Application processesA1 On block ID:3
6.) Application processesA2 On block ID:3
7.) Application processesA3 On block ID:3
8.) Application processesA1 On block ID:3
9.) Application processesA2 On block ID:3
```

1.4.2 Test case 2

Input.txt

```
{
  "1": {
    "Application": {
      "MiddlewareUp": {
        "port": 3000,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3001,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3000,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3001,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5000,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5001,
        "host": "localhost"
      }
    }
  },
  "2": {
    "Application": {
      "MiddlewareUp": {
        "port": 3002,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3003,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3002,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3003,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5002,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5003,
        "host": "localhost"
      }
    }
  },
  "3": {
    "Application": {
      "MiddlewareUp": {
        "port": 3004,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3005,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3004,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3005,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5004,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5005,
        "host": "localhost"
      }
    }
  },
  "4": {
    "Application": {
      "MiddlewareUp": {
        "port": 3006,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3007,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3006,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3007,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5006,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5007,
        "host": "localhost"
      }
    }
  }
}
```

Outputs:

1.txt

```
1.) Application processesA2 On block ID:1
2.) Application processesA3 On block ID:1
3.) Application processesA2 On block ID:1
4.) Application processesA2 On block ID:1
5.) Application processesA3 On block ID:1
6.) Application processesA3 On block ID:1
7.) Application processesA2 On block ID:1
```

2.txt

1.) Application processesA2 On block ID:2
2.) Application processesA3 On block ID:2
3.) Application processesA2 On block ID:2
4.) Application processesA2 On block ID:2
5.) Application processesA3 On block ID:2
6.) Application processesA3 On block ID:2

3.txt

1.) Application processesA2 On block ID:3
2.) Application processesA3 On block ID:3
3.) Application processesA2 On block ID:3
4.) Application processesA2 On block ID:3
5.) Application processesA3 On block ID:3
6.) Application processesA3 On block ID:3

4.txt

1.) Application processesA2 On block ID:4
2.) Application processesA3 On block ID:4
3.) Application processesA2 On block ID:4
4.) Application processesA2 On block ID:4
5.) Application processesA3 On block ID:4
6.) Application processesA3 On block ID:4
7.) Application processesA2 On block ID:4

1.4.3 Test case 3

Input.txt

```
{
  "1": {
    "Application": {
      "MiddlewareUp": {
        "port": 3000,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3001,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3000,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3001,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5000,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5001,
        "host": "localhost"
      }
    }
  },
  "2": {
    "Application": {
      "MiddlewareUp": {
        "port": 3002,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3003,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3002,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3003,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5002,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5003,
        "host": "localhost"
      }
    }
  },
  "3": {
    "Application": {
      "MiddlewareUp": {
        "port": 3004,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3005,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3004,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3005,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5004,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5005,
        "host": "localhost"
      }
    }
  },
  "4": {
    "Application": {
      "MiddlewareUp": {
        "port": 3006,
        "host": "localhost"
      },
      "MiddlewareDown": {
        "port": 3007,
        "host": "localhost"
      }
    },
    "Middleware": {
      "ApplicationUp": {
        "port": 3006,
        "host": "localhost"
      },
      "ApplicationDown": {
        "port": 3007,
        "host": "localhost"
      },
      "NetworkUp": {
        "port": 5006,
        "host": "localhost"
      },
      "NetworkDown": {
        "port": 5007,
        "host": "localhost"
      }
    }
  }
}
```

Output.txt

1.txt

```
1.) Application processesA1 On block ID:1
2.) Application processesA1 On block ID:1
3.) Application processesA1 On block ID:1
4.) Application processesA4 On block ID:1
5.) Application processesA1 On block ID:1
6.) Application processesA2 On block ID:1
7.) Application processesA4 On block ID:1
8.) Application processesA1 On block ID:1
```

2.txt

```
1.) Application processesA1 On block ID:2
2.) Application processesA1 On block ID:2
3.) Application processesA1 On block ID:2
4.) Application processesA4 On block ID:2
5.) Application processesA1 On block ID:2
6.) Application processesA2 On block ID:2
7.) Application processesA4 On block ID:2
8.) Application processesA1 On block ID:2
```

3.txt

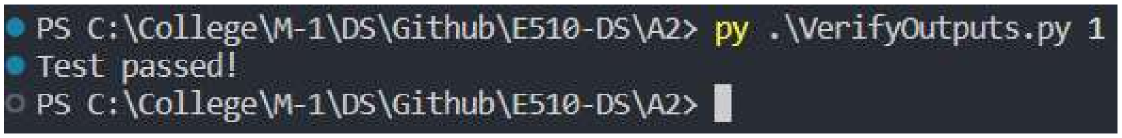
```
1.) Application processesA1 On block ID:3
2.) Application processesA1 On block ID:3
3.) Application processesA1 On block ID:3
4.) Application processesA4 On block ID:3
5.) Application processesA1 On block ID:3
6.) Application processesA2 On block ID:3
7.) Application processesA4 On block ID:3
8.) Application processesA1 On block ID:3
```

4.txt

1.) Application processesA1 On block ID:4
2.) Application processesA1 On block ID:4
3.) Application processesA1 On block ID:4
4.) Application processesA4 On block ID:4
5.) Application processesA1 On block ID:4
6.) Application processesA2 On block ID:4
7.) Application processesA4 On block ID:4
8.) Application processesA1 On block ID:4

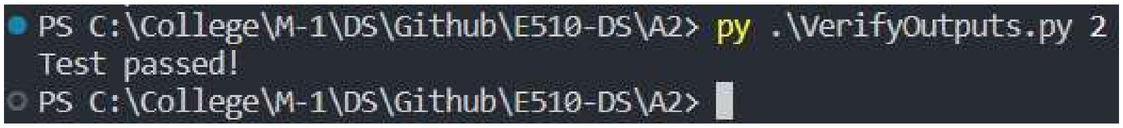
1.5 Output Verification

1.5.1 Test Case 1:



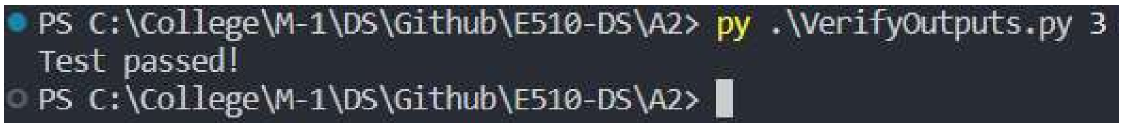
```
PS C:\College\M-1\DS\Github\E510-DS\A2> py .\VerifyOutputs.py 1
Test passed!
PS C:\College\M-1\DS\Github\E510-DS\A2>
```

1.5.2 Test Case 2:



```
PS C:\College\M-1\DS\Github\E510-DS\A2> py .\VerifyOutputs.py 2
Test passed!
PS C:\College\M-1\DS\Github\E510-DS\A2>
```

1.5.2 Test Case 3:



```
PS C:\College\M-1\DS\Github\E510-DS\A2> py .\VerifyOutputs.py 3
Test passed!
PS C:\College\M-1\DS\Github\E510-DS\A2>
```

1.6 Limitations

The limitations of the system so implemented are as follows:

1. Due to each block running on different processors, multi-processing is the main issue. Where we cannot test with multiple clients because of the processor limitations. So if we have a system with multiple processors it can be tested thoroughly.
2. Currently, the system only has a particular kind of message that is generated but we can have users provide the input messages and then process it to test it thoroughly.
3. The system cannot entertain the user that would join after the spawning.

1.7 References

[1] Socket.io: https://python-socketio.readthedocs.io/en/stable/
[2] Threading: https://docs.python.org/3/library/threading.html
[3] Multiprocessing: https://docs.python.org/3/library/multiprocessing.html
[4] OS module: https://docs.python.org/3/library/os.html
[5] Princeton paper:
https://www.cs.princeton.edu/courses/archive/spring21/cos418/docs/L6-vc.pdf
[6] Github Repo: https://github.com/jenilgandhi2111/E510-DS