MONASH University

ASSIGNMENT COVER SHEET

Student's name	(Family name) Liu	(Given names) Yang	
ID number	28276345	**E-mail**	Yliu0494@student.monash.edu
Unit code & name	FIT3143 Parallel Computing	**Unit code**	3143

Title of assignment	**Assignment – 2** ***(You may include an appropriate title as per your report)***
Lecturer/tutor	

Is this an authorised group assignment? ☐ **Yes** ☒ **No**

If this submission is a group assignment, each student must attach their own signed cover sheet to the assignment.

Has any part of this assignment been previously submitted as part of another unit/course? ☐ **Yes** ☒ **No**

Tutorial/laboratory day & time	Tuesday 7-9pm
Due date: 18/10/2023	**Date submitted: 18/10/2023**

All work must be submitted by the due date. If an extension of time to submit work is required, a Special Consideration Application (In-semester Assessment Task) must be submitted.

Has an extension been approved? **Yes** ☑ **No** ☐ **If yes, please give the new submission date : 18/10/2023**

Please note that it is your responsibility to retain copies of your assessments.

Student Statement:

- I have read the University's Student Academic Integrity Policy and the University's Student Academic Integrity: Managing Plagiarism and Collusion Procedures.
- I understand the consequences of engaging in plagiarism and collusion as described in Assessment and Academic Integrity Policy
- I have taken proper care of safeguarding this work and made all reasonable effort to ensure it could not be copied.
- I acknowledge that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  i. provide to another member of faculty; and/or
  ii. submit it to a plagiarism checking service; and/or
  iii. submit it to a plagiarism checking service which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Signature .....Yang Liu..... *Date.....18/10/2023.....*

Privacy Statement

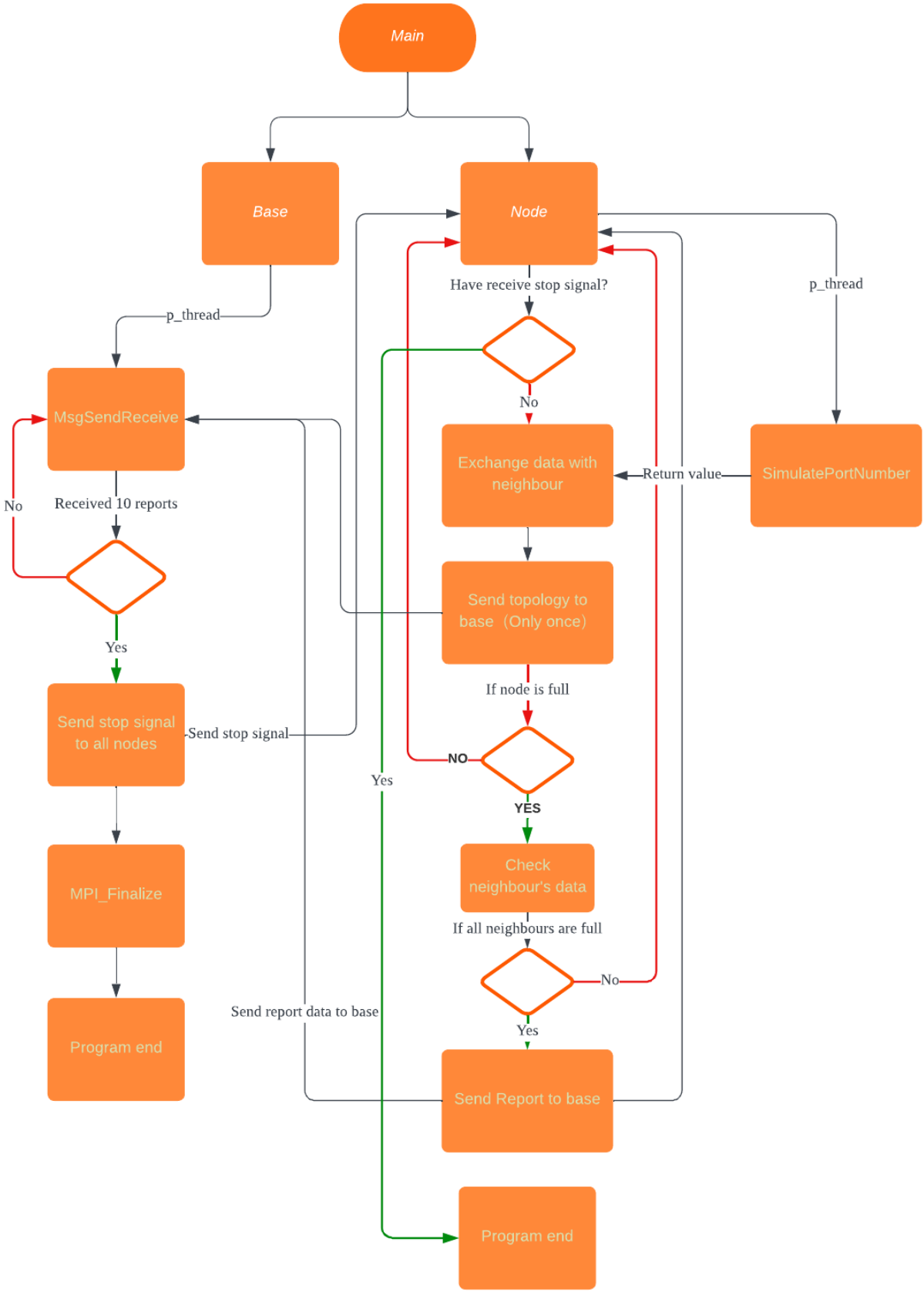
The information on this form is collected for the primary purpose of assessing your assignment. Other purposes of collection include recording your plagiarism and collusion declaration, attending to course and administrative matters and statistical analyses. If you choose not to complete all the questions on this form it may not be possible for Monash University to assess your assignment. You have a right to access personal information that Monash University holds about you, subject to any exceptions in relevant legislation. If you wish to seek access to your personal information or inquire about the handling of your personal information, please contact the University Privacy Officer: privacyofficer@adm.monash.edu.au

FIT3143 Semester 2, 2023
Assignment 2 – Report
Title – Specify your tile based on the Assignment Specifications

Note: Please refer to Assignment specifications, FAQ and marking guide for details to be included in the following sections of this report.

Include the word count here (for Sections A to C): _605___

A. Methodology



MONASH University

Code Description:

Main() Function:
In the `main` function, the program initializes MPI with multiple threads. It checks user input for the number of rows and columns, ensuring it matches the number of threads minus one. The last thread is designated as the master thread (base_io()), while the rest are considered nodes (`node_io()`). The communicator is split into `MPI_COMM_WORLD` for all threads and `new_comm` for nodes. Nodes communicate with the base using `MPI_COMM_WORLD` and with their neighbors using `new_comm`.

base_io() function
The thread with the highest rank executes this function. It creates a POSIX thread and runs the `MsgSendReceive()` function. `MsgSendReceive()` is responsible for receiving and processing messages. The function extracts necessary information from the provided thread data structure and MPI communicator (`MPI_COMM_WORLD`). It enters a loop, receiving messages from different sources via `MPI_Recv`. Depending on the message tag, various actions are taken. The loop stops after printing 10 reports and receiving exit signals from all nodes.

1. Messages with the 'MSG_EXIT' tag decrement the number of active nodes.
2. Messages with the 'MSG_INITIALIZE' tag initialize the topology information for nodes. Each node's neighbors are received, and the data is stored in the 'topology' variable. For instance, `topology[0] = {-2, 1, -2, 3}` indicates node 0 with neighbors 1 and 3 (or -2 for no neighbor).
3. Messages with the 'MSG_REPORT' tag process report data. When a node and its neighbors are full, the reporting node sends a string to the base with tag 'MSG_REPORT'. The data, separated by commas, is processed to generate a txt log file.

node_io() function
This function creates a 2D Cartesian communicator (`comm2D`) and extracts information about the node's rank, coordinates, and neighbors. It continuously checks for termination signals from the base process. Upon receiving a termination signal (with tag 'MSG_END'), the function sets `termination_received` to 1, indicating the node should terminate. It starts by simulating port usage using the `SimulatePortNumber()` function via a pthread, returning the value to the variable `num`. The function also receives an initialization message from the master process (with tag 'MSG_INITIALIZE') and responds with the node's topology information using the same tag. When the node's port availability drops below a threshold (`K - 2`), it sends a report to the master process (with tag 'MSG_REPORT') containing the node's rank, coordinates, port availability, and neighboring nodes' information. The function includes a `sleep(1)` statement, pausing the node's execution for 1 second in each iteration. The loop continues until a termination signal is received.

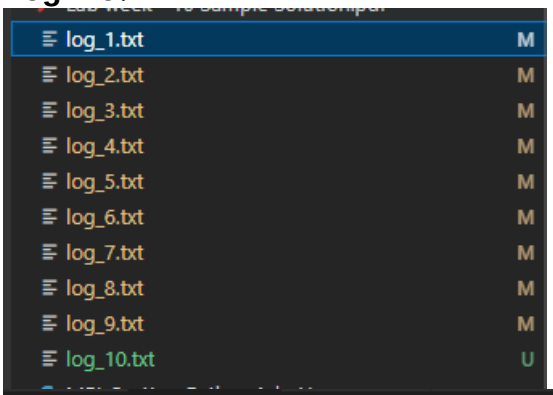
SimulatePortNumber() Function:
This function generates random numbers for each rank, ranging from 0 to K.

B. Results Tabulation

Topology of 9 nodes

```
student@f14483fb40bf:~/project/3143a2$ mpirun -np 10 -oversubscribe a2 3 3
Show topology of node number: 5, neighbours: 4 -2 2 8
Show topology of node number: 2, neighbours: 1 -2 -2 5
Show topology of node number: 1, neighbours: 0 2 -2 4
Show topology of node number: 0, neighbours: -2 1 -2 3
Show topology of node number: 3, neighbours: -2 4 0 6
Show topology of node number: 6, neighbours: -2 7 3 -2
Show topology of node number: 8, neighbours: 7 -2 5 -2
Show topology of node number: 7, neighbours: 6 8 4 -2
Show topology of node number: 4, neighbours: 3 5 1 7
Report number: 1 Node: 2 reports full
Report number: 2 Node: 2 reports full
Report number: 3 Node: 8 reports full
Report number: 4 Node: 8 reports full
Report number: 5 Node: 6 reports full
Report number: 6 Node: 2 reports full
Report number: 7 Node: 8 reports full
Report number: 8 Node: 6 reports full
Report number: 9 Node: 8 reports full
Report number: 10 Node: 6 reports full
Stop signal send to node: 0
Stop signal send to node: 1
Stop signal send to node: 2
Stop signal send to node: 3
Stop signal send to node: 4
Stop signal send to node: 5
Stop signal send to node: 6
Stop signal send to node: 7
Stop signal send to node: 8
Boardcast finished
master finished
student@f14483fb40bf:~/project/3143a2$
```

Log file：



log_1.txt

```
1   Max port number 10
2   Report number 1
3   Alert reported time : 2023-10-18 08:52:52
4   Logged time: 2023-10-18 08:52:52
5   Reporting node      Coordinate      Prot Value      Availability to be considered full
6   2                   (0,2)           8               2
7   Adjacent node       Coordinate      Prot Value      Availability to be considered full
8   1                   (0,1)           9               1
9   5                   (1,2)           8               2
10  Nearby Nodes        Coordinate
11  0                   (0,0)
12  4                   (1,1)
13  8                   (2,2)
14  Available station nearby (no report received in last 3 iteration): 0 4 8
15  End of file
16
```

log_2.txt

```
Max port number 10
Report number 2
Alert reported time : 2023-10-18 08:53:33
Logged time: 2023-10-18 08:53:33
Reporting node     Coordinate     Prot Value     Availability to be considered full
2                  (0,2)          9              1
Adjacent node      Coordinate     Prot Value     Availability to be considered full
1                  (0,1)          8              2
5                  (1,2)          9              1
Nearby Nodes      Coordinate
0                 (0,0)
4                 (1,1)
8                 (2,2)
Available station nearby (no report received in last 3 iteration): 0 4 8
End of file

```

log_3.txt

```
Max port number 10
Report number 3
Alert reported time : 2023-10-18 08:53:41
Logged time: 2023-10-18 08:53:41
Reporting node     Coordinate     Prot Value     Availability to be considered full
8                  (2,2)          8              2
Adjacent node      Coordinate     Prot Value     Availability to be considered full
7                  (2,1)          9              1
5                  (1,2)          8              2
Nearby Nodes      Coordinate
2                 (0,2)
4                 (1,1)
6                 (2,0)
Available station nearby (no report received in last 3 iteration): 4 6
End of file

```

CAAS:

```
[yliu0494@student-caas-headnode 3143a2]$ cat parallel.13524.out
nrows 0, ncols 72, cores: 1
Boardcast finished
master finished
```

C. Analysis & Discussion

The master thread (base_io) communicates with slave threads (node_io) using the MPI_COMM_WORLD communicator. It sends initialization messages (MSG_INITIALIZE) to all slave threads to collect topology information.

This topology information is crucial for generating reports and coordinating charging nodes' activities. Upon receiving a report message, the master thread processes the data, including the reporting node's rank, coordinates, port availability, and information about neighboring nodes.

This communication flow ensures coordinated execution between the master and slave threads, enabling efficient charging node management and report generation in the parallel computing environment.

Due to the inability to obtain the log files correctly, I am unable to obtain the results of running on the large-scale cluster on CAAS

D. References

Pass values in pthread
https://stackoverflow.com/questions/1352749/multiple-arguments-to-function-called-by-pthread-create