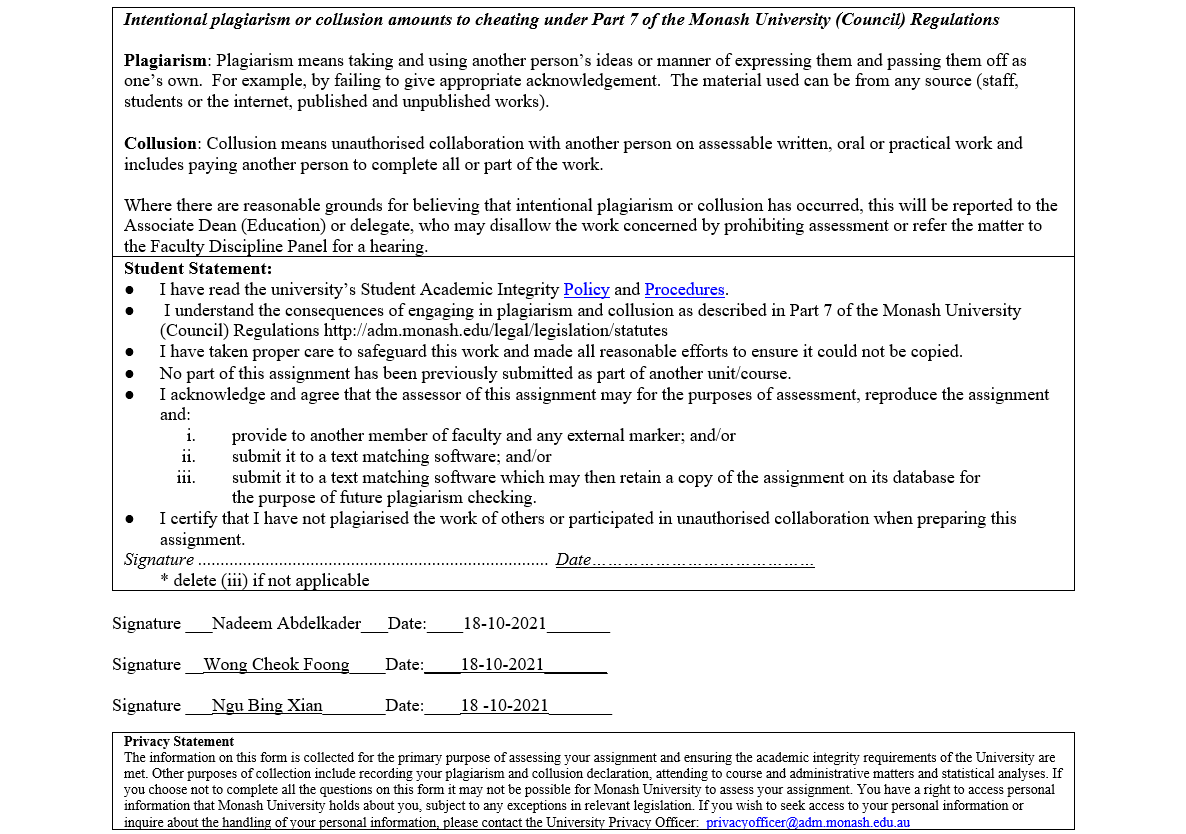
Table

Description automatically generated

**FIT3143 Semester 2, 2021**

**Assignment 2 - Report**

**Team Name (or Number):** MA\_2021\_LAB-06\_Team\_09

| **Student email address** | **Student First Name** | **Student Last Name** | **Contribution %** | **Contribution details\*** |
| --- | --- | --- | --- | --- |
| cwon0031@student.  monash.edu | Cheok Foong | Wong | 33.33% | Worked on the sensor nodes for the code.  Worked on the sensor nodes for report methodology.  Worked on parts of the tabulation and parts of the analysis for the report. |
| nabd0018@student.  monash.edu | Nadeem Emadeldin | Abdelkader | 33.33% | Worked on the satellite altimeter & attempted to implement the fault detection (Code & Report methodology). Worked on parts of the tabulation. Did all flow/bar charts & grid representation of nodes. |
| bngu0011@student.  monash.edu | Bing Xian | Ngu | 33.33% | Worked on the base station.  Worked on some of the parts in result & tabulation and analysis & discussion section |

*\*Your contribution details include the report, code, or both.*

Include the word count here (for Sections A to C): **(A: 1586, B: 24, C: 261)**

**Simulate a distributed wireless sensor network to detect possible tsunamis**

Ngu Bing Xian (bngu0011@student.monash.edu)

Wong Cheok Foong (cwon0031@student.monash.edu)

Nadeem Abdelkader (nabd0018@student.monash.edu)

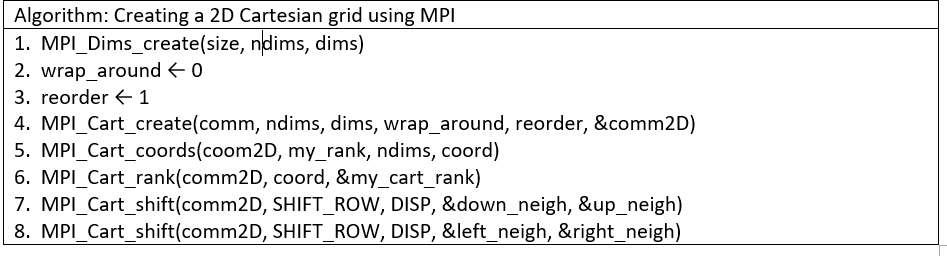
**Word Count**: 1871 words

**A: Methodology**

**A.1 Sensor Node**

**Structure of sensor nodes**

The entire structure of the sensor nodes are being implemented by creating a 2D Cartesian grid using MPI virtual topology. The architecture is designed in a way where each particular node is able to communicate with its adjacent nodes which are the top, bottom, left and right nodes. The grid layout being used by the WSN is specified by the user. If the number of processors being used by the nodes in the system is 6 then the possible grid layout can either be  [2 x 3], [3 x 2], [1 x 6] or [6 x 1].



The algorithm above shows how a 2D cartesian grid is created using MPI for the sensor nodes. The **MPI\_Dims\_create** creates a dimension grid based on the specified ndims values which we specify as 2 in the system since we want a 2D grid. The wrap\_around indicates whether periodic shifting occurs and 0 means it’s false. The reorder indicates that reordering of rankings is true. The **MPI\_Cart\_create** makes a new communicator to which topology information has been attached. The **MPI\_Cart\_coords** allows each rank to find their own coordinates in the cartesian group. The **MPI\_cart\_rank** allows each coordinate to find their own rank in the cartesian group. **MPI\_Cart\_shift** allows each node to know who its adjacent nodes are, which is 1 displacement for up, down, left and right.

Chart

Description automatically generated

*Figure 1: 4 x 4 grid layout*

The figure provided above which is a [4 x 4] grid layout consisting of 16 sensor nodes shows the exact structure of how the sensor nodes are positioned. The 1st node starts from the top left corner and then populates from left to right. Using node 11 which is being marked red as an example, we can see that it has four adjacent nodes which are 7, 10, 12 and 15 which are being marked green and can communicate with them to acquire seawater column height value (SWCHV) for comparison. However, we can also see that node 1 only has 2 adjacent nodes which are nodes 2 and 5 as its top and left nodes are non-existent. This is because the grid is not designed in such a way where periodic shifts occur which means there is no wrap-around for each edge node.

**Communication between each sensor node and base station**

Each sensor node will consistently generate a random SWCHV which will then be stored in a fixed array size. Once the array is full then the simple moving average (SMP) is calculated and this value is then sent to each of its adjacent nodes using **MPI\_Isend** and then each of the adjacent nodes will receive the value using **MPI\_Irecv.** Each SMP that exceeds the predefined threshold value will then be compared with the SMP values of its adjacent nodes which have just been received to compare similar readings. If >= 2 adjacent nodes have similar readings then a report containing required information will be sent to the base station.

The required information is being stored in a struct formatusing **MPI\_Type\_create\_struct** which allows combining data of different types and also **MPI\_Type\_commit** to commit the newly created data type for the structure. A struct object containing the pieces of information will then be sent to the base station using **MPI\_Send**. Figure 2 below shows a diagram of the list of data items to be sent to the base station.

A picture containing graphical user interface

Description automatically generated

*Figure 2: report structure*

The node coordinates and report datetime are needed to compare and determine a match reading with the satellite altimeter. The rest of the reported details are needed to be written to the output file as key informations.

**Termination of each sensor node**

Each sensor node will keep on producing the SWCHV infinitely in a while loop and only breaks when it receives an exit message from the base station using **MPI\_Iprobe** which constantly checks for any incoming messages and uses **MPI\_Recv** once an exit message is detected. A constant tag **EXIT\_TAG** is defined for the sensor nodes to recognize that an exit message has been sent with the particular tag. After receiving the message each node will break the while loop which then terminates the process.

**A.2 Satellite Altimeter**

The satellite altimeter is simulated as a single posix threat that the base station creates.

Its main aim is to periodically generate random readings that exceed the specified threshold. In addition, it generates random coordinates within the range of the cartesian grid layout, which was specified earlier.

**The following is the data the satellite altimeter generated periodically.**

[Year (YYYY), Month (MM), Day (DD), Hour (HH), Minutes (MM), Seconds (SS), Random coordinates within the the cartesian grid layout, Random readings that exceed the predefined threshold]

**Example:**

[2021, 10, 17, 6, 13, 45, (3,2), 6400.215]

This data is stored in a shared array that can be accessed by the base station

The shared array is of a fixed size and follows a queue like approach (first in first out)

This process will continue on repeatedly until the satellite altimeter receives a termination signal from the base station. When this happens the thread will exit.

Diagram

Description automatically generated

*Figure 3: Satellite altimeter populating shared array with random readings that exceed the threshold*

The diagram above (figure 3) is a simple flow chart showing the satellite altimeter main process

**A.3 Base station**

*3.0 Introduction*

Base station will be divided into two communicators, which are the master and slave. Master creates 3 threads which have 3 separate tasks that will be carried out by each thread, while slave is responsible for creating the Sensor nodes.

*3.1 Master*

Base station creates three threads that has its respective responsibilities based on its rank, which are

1. Thread 1: Produce summary log, shutdown Altimeter and Sensor Node
2. Thread 2: Altimeter - generates random coordinates and sea water column height
3. Thread 3: Listen to messages from Wireless Sensor Node (WSN), compare the values of generated by the altimeter and the messages sent by WSN

*3.1.1 Thread 1*

In this thread, it is used to produce a summary log, shutdown the Sensor node and the Altimeter when the iteration ends. **MPI\_Send** is used to send the termination message to the Sensor node, together with **EXIT\_TAG** to indicate that the message sent is for termination of the sensor nodes.

*3.1.2 Thread 2*

This section has been explained in “Subsection B: Satellite Altimeter”

*3.1.3 Thread 3*

In this thread, it has a responsibility of constantly listening to the message sent by the WSN and then comparing the message sent to see if there is a match of the coordinate in the array produced by the Altimeter within a certain time period.

**MPI\_Iprobe** is used to constantly listening to messages sent by WSN. It takes in 5 arguments, which are

1. source of the message,
2. a tag
3. communication
4. a flag
   1. flag is used to indicate when there is a message that is being sent by WSN
5. the status
   1. status can be used to identify the source of the message by using status.MPI\_SOURCE.

**MPI\_Recv** is used to receive the message that is listened to by the MPI\_Iprobe.

Graphical user interface, text, application, email

Description automatically generated

Algorithm 1 is used for verifying the message received from the Sensor Node and the Altimeter Shared Array reading. In this algorithm, it will first loop the entire list of the shared array to check if there is a match between the elements in the list and the coordinate of the sensor node. It will also check for the time between both sensor nodes and the matched elements is within 10 seconds to ensure the correctness of the report. If both criteria are met, it will then check if the difference between the height of the sensor node and the matched element is within a threshold. A true alert will be assigned if the height is within the threshold and false otherwise. A result will be returned after the execution of the verify function to be logged into a report file.

*3.2 Slave*

This section is responsible for creating Sensor nodes, which has been explained in “Subsection A: Sensor node”.

**A.4 Fault detection**

Using MPI communicators and virtual topologies together with a thread to allow the master node to check the alive status of the slave nodes within a group.

Master node keeps track of the last active status of the slave nodes.

Passive approach of checking the alive status of the slave/worker nodes.

Slave nodes periodically send “alive” messages to the master node or “exit” message if the slave  node has completed its task and is exiting.

The main process application will spawn a thread that listens for “alive” messages from the slave nodes, while the main thread continues its process normally.

Main thread will also listen to exit messages from the nodes, when it receives exit messages for all slave nodes it will signal the thread to exit and once the thread has received the signal it then returns and joins to the main process and then the main process exits gracefully.

The main process will use MPI\_Probe and MPI\_RECV to listen for messages with particular tags (report tag) and the spawned thread will do the same thing but to listen for the alive tag instead.

When the thread receives the alive messages from the slave nodes it will calculate the last active/alive time.

The last active time can be used to determine if there is a faulty node or nodes by setting an acceptable threshold for the last active time. In other words, if the last active time for a certain node is outside the predefined threshold we interpret that as a faulty node.

Diagram

Description automatically generated

*Figure 4: Fault detection flow chart*

**B. Result Tabulation**

**Platform tested on**: Virtual machine

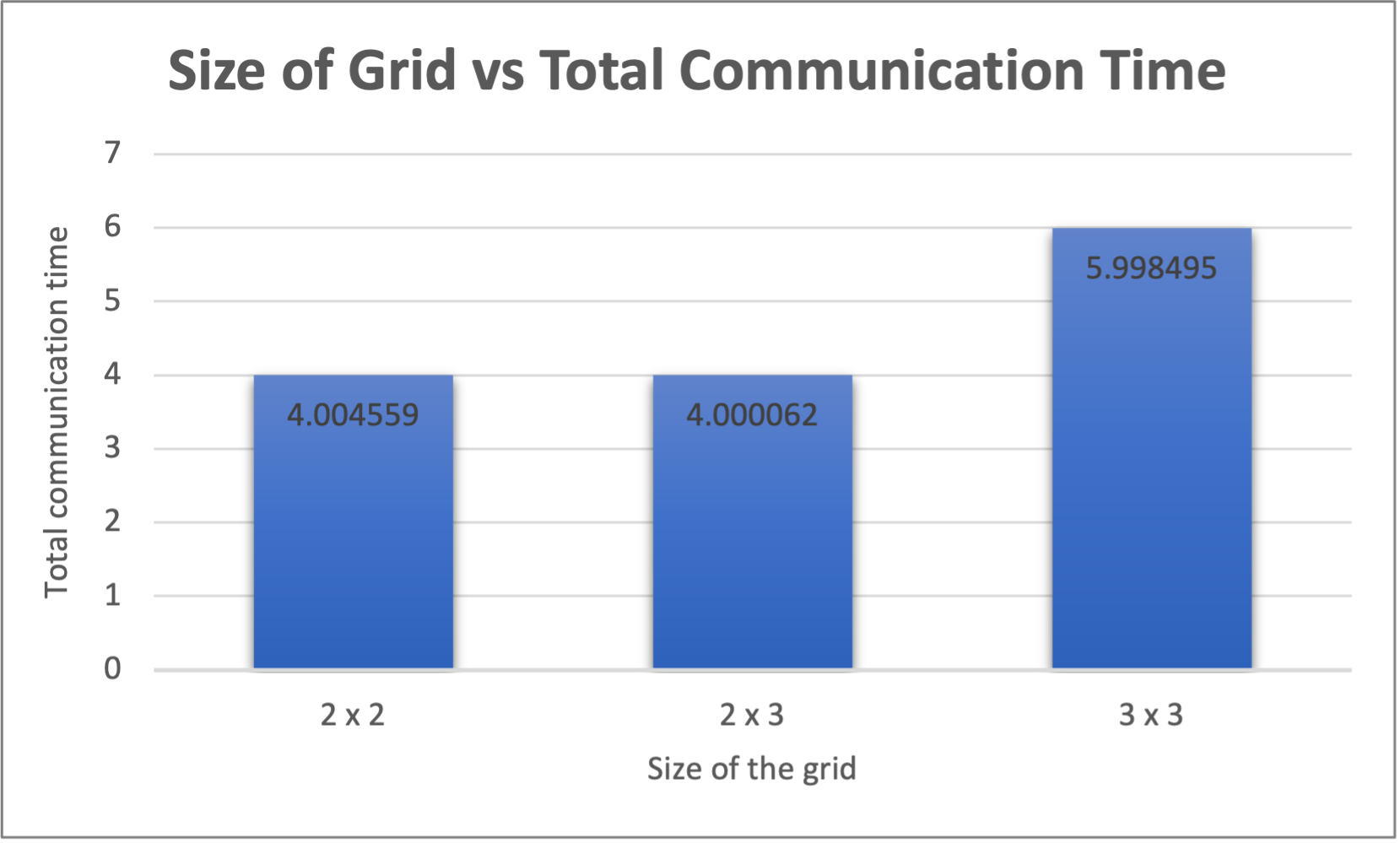
**Specifications of the platform:**

| **Specification** | **Value** |
| --- | --- |
| Logical processors | 8 |
| System memory | 4098MB |
| CPU frequency | 3.16Ghz |

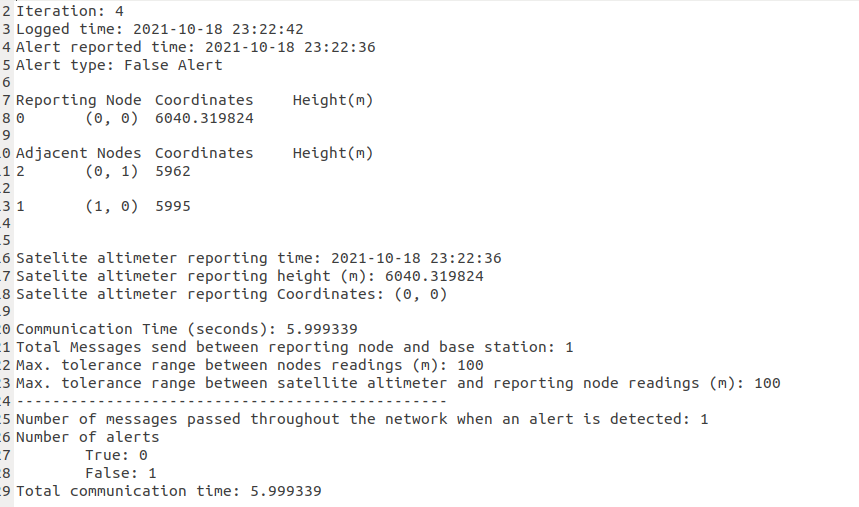
**Specification of each test run:**

Performance of the sensor node for different number of grid sizes

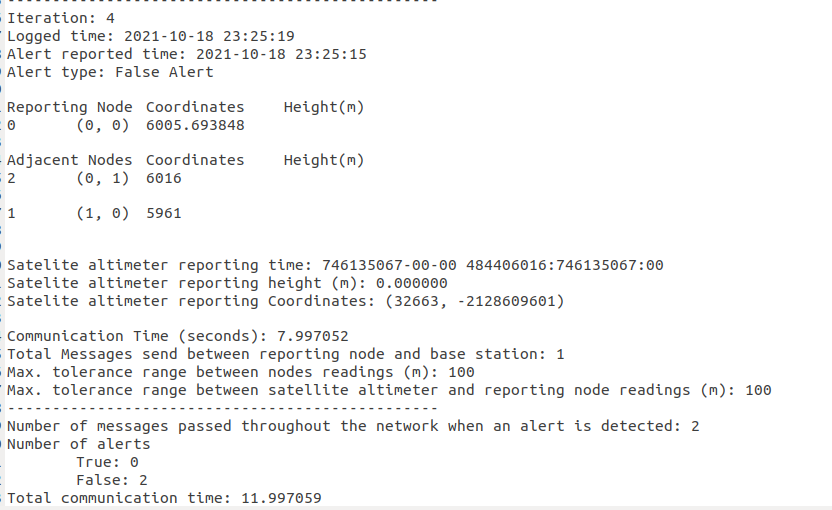
|  | Size of the grid | Sea water column height threshold | Number of iterations at the base station | Total communication time(s) |
| --- | --- | --- | --- | --- |
| Test 1 | 2 x 2 | 6000 | 5 | 5.999339 |
| Test 2 | 2 x 3 | 6000 | 5 | 11.997059 |
| Test 3 | 3 x 3 | 6000 | 5 | 6.015579 |



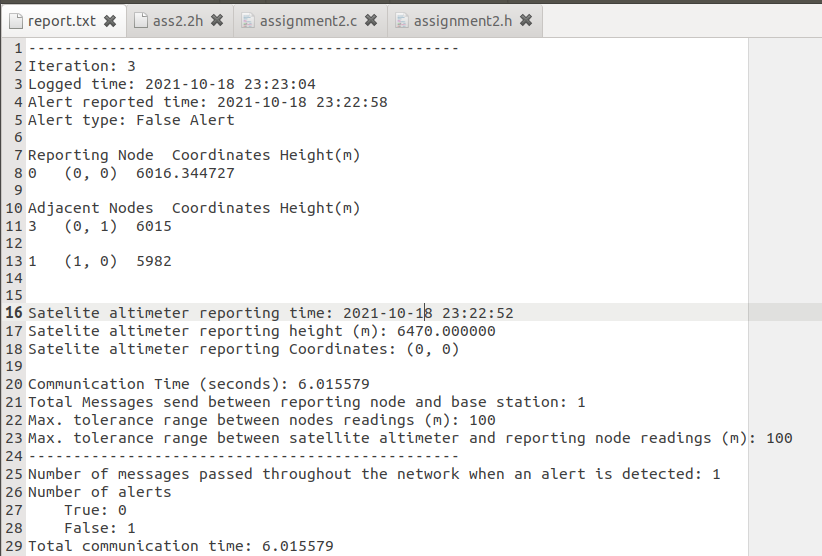
**Screenshots of log1.txt file for 2 x 2 grid**



**Screenshot of log2.txt file for 2 x 3 grid**

****

**Screenshot of log3.txt file for 3 x 3 grid**

****

**C. Analysis & Discussion**

**Hypothesis**

For a given grid size, the communication time between the sensor nodes and the base station increases non-linearly. According to the design of our program, the sensor nodes only sends a report message to the base station if the readings of the sea water column height (SWCH) exceeds a predefined threshold and does not always send for every SWCH it produces. Assuming that the number of cycles required to calculate the simple moving average and also the tolerance range between the sensor nodes for each test run is the same, sometimes the sensor nodes may take a longer time to send a report message as it not only needs to wait for the SWCH to exceed the threshold but also has to compare with its adjacent nodes to check for similar readings and confirm the alert and vice versa. This is why the total communication time between the sensor nodes and the base station varies even with an increasing number of grid sizes.

**Limitation**

Since there is a limited number of processes in a virtual machine, hence, the size of the grid has been limited to a smaller grid as compared to the number of processes that Monarch can support. For example, the maximum number of processes a virtual machine can support is 8, hence, the size of the grid of our test runs is limited to less or equal to 8. Although using -oversubscribe can be a way of running more than the limited number of processes a virtual machine can support, the communication time between the processes might be affected.

**D. References**

Kendall, W. (2021). *Dynamic Receiving with MPI Probe (and MPI Status)*. MPI Tutorial. Retrieved October 17, 2021, from https://mpitutorial.com/tutorials/dynamic-receiving-with-mpi-probe-and-mpi-status/

Declaration:

I declare that this assignment report and the submitted code represent work within my team. I have not copied from any other teams’ work or from any other source except where due acknowledgment is made explicitly in the report and code, nor has any part of this submission been written for me by another person outside of my team.

Signature of student 1:             \_\_\_\_ Wong Cheok Foong\_\_\_\_\_\_\_\_\_

Signature of student 2:             \_\_\_\_Ngu Bing Xian\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of student 3:             \_\_\_\_Nadeem Abdelkader\_\_\_\_\_\_\_\_\_