

## GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
29940672	Tah	Wen Zhong
29938767	Hor	Ethan, Sheng Jian

\* Please include the names of all other group members.

<b>Unit name and code</b>	FIT3143 Parallel computing		
<b>Title of assignment</b>	Assignment 2		
<b>Lecturer/tutor</b>	Shageenderan Sapai		
<b>Tutorial day and time</b>	Friday 9am – 11am	<b>Campus</b>	Monash University
<b>Is this an authorised group assignment?</b> <input checked="" type="checkbox"/> Yes <input type="checkbox"/> No			
<b>Has any part of this assignment been previously submitted as part of another unit/course?</b> <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No			
<b>Due Date</b>	18/10/2021	<b>Date submitted</b>	18/10/2021

All work must be submitted by the due date. If an extension of work is granted this must be specified with the signature of the lecturer/tutor.

**Extension granted until (date)** ..... **Signature of lecturer/tutor** .....

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

### ***Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations***

**Plagiarism:** Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

**Collusion:** Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.


#### **Student Statement:**

- I have read the university's Student Academic Integrity [Policy](#) and [Procedures](#).
- I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations <http://adm.monash.edu/legal/legislation/statutes>
- I have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
- No part of this assignment has been previously submitted as part of another unit/course.
- I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  - i. provide to another member of faculty and any external marker; and/or
  - ii. submit it to a text matching software; and/or
  - iii. submit it to a text matching software 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** ..... **Date** .....

\* delete (iii) if not applicable

Signature  Date: 18/10/2021 Signature \_\_\_\_\_ Date: \_\_\_\_\_

Signature  Date: 18/10/2021 Signature \_\_\_\_\_ Date: \_\_\_\_\_

Signature \_\_\_\_\_ Date: \_\_\_\_\_ Signature \_\_\_\_\_ Date: \_\_\_\_\_

#### **Privacy Statement**

The information on this form is collected for the primary purpose of assessing your assignment and ensuring the academic integrity requirements of the University are met. 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](mailto:privacyofficer@adm.monash.edu.au)

## FIT3143 Semester 2, 2021 Assignment 2 - Report

**Team Name (or Number):** FIT3143\_S2\_2021\_MA\_LAB-04\_Team\_03

Student email address	Student First Name	Student Last Name	Contribution %	Contribution details*
wtah0001@student.monash.edu	Wen Zhong	Tah	50	both
ehor0005@student.monash.edu	Ethan, Sheng Jian	Hor	50	both

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

Note: Please refer to Assignment [specifications](#), [FAQ](#) and marking rubric ([two member](#) or [three member](#) teams) for details to be included in the following sections of this report.

Include the word count here (for Sections A to C): \_\_\_\_1411\_\_\_\_

### A. Methodology

The system we devised utilizes multithreaded processes for the simulation of a WSN with a set of interconnected sensors.

#### Grid architecture

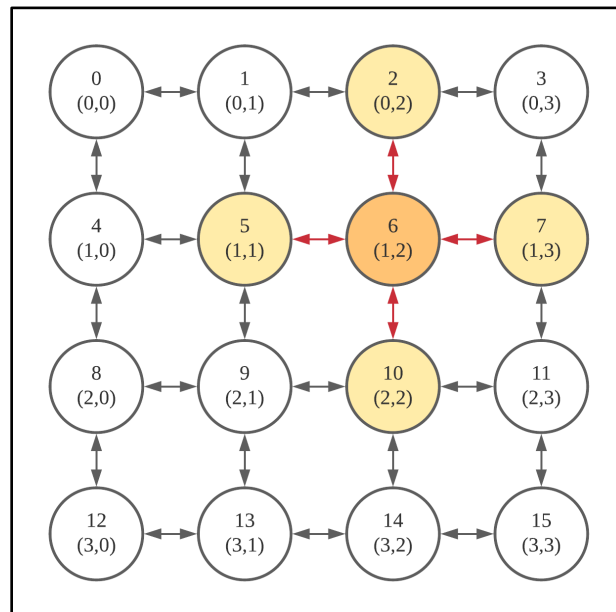


Figure 1: Cartesian grid layout

The WSN follows a cartesian grid layout, with one MPI process simulating each node. As such, we used MPI virtual topology to map the MPI process into a Cartesian virtual topology. Figure 1 illustrates an example of one such mapping for a 4 x 4 grid layout. Each node will only communicate with the base station and its adjacent nodes. For instance, the rank 6 node in Figure 1 can only communicate with the adjacent nodes highlighted, that is node 2, node 10, node 5 and node 7.

## Initializing the Wireless Sensor Network

During initialization, a new MPI datatype will be created from a set of data types which will be the report sent by the sensor nodes to the base station. Following this, a new communicator will be created through splitting the default communicator. This subcommunicator contains all processes dedicated towards simulating the sensor nodes. In our system, only the rank 0 MPI process will be excluded as it simulates the base station. In addition, this will also create a POSIX thread which simulates the satellite altimeter.

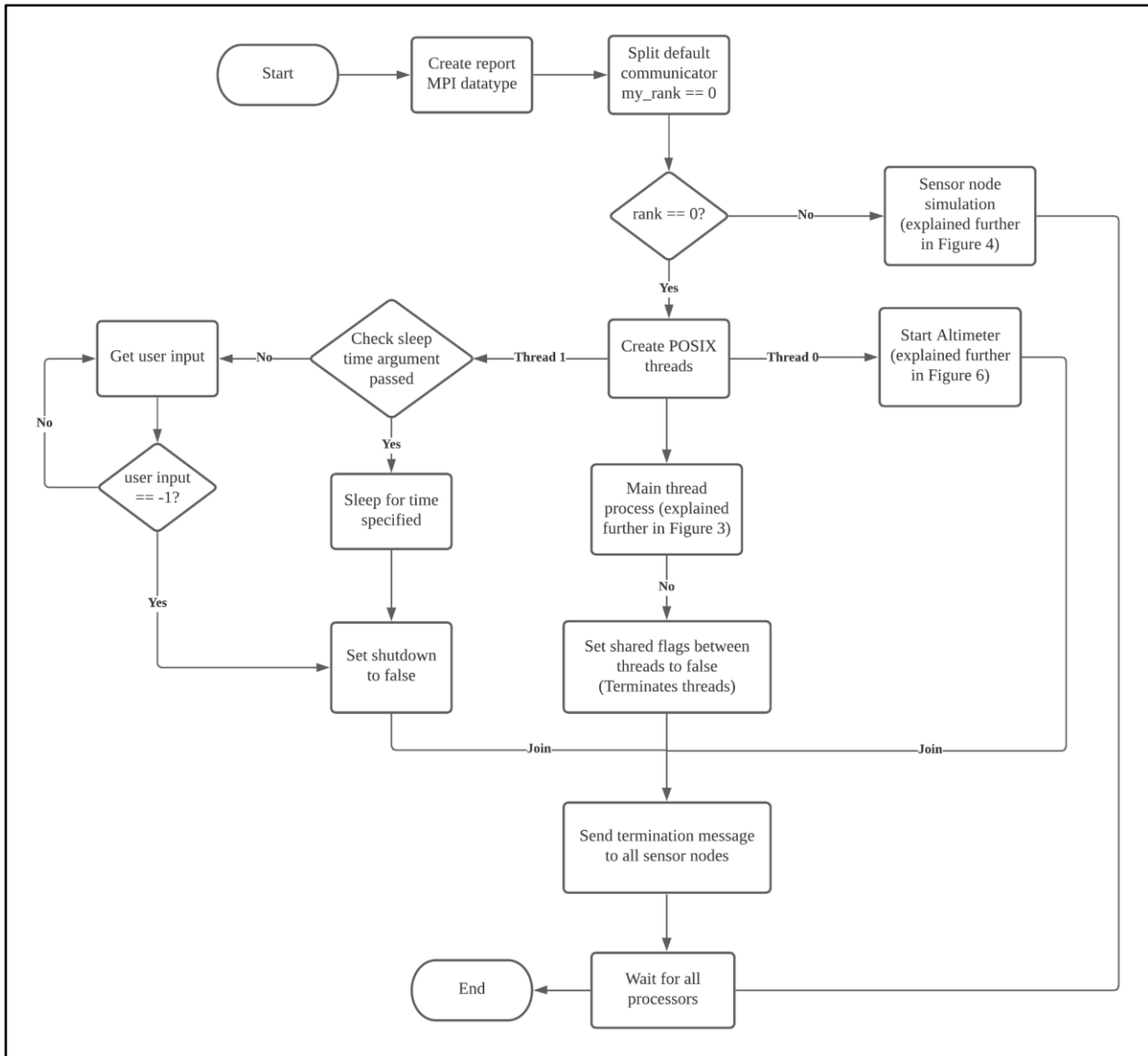


Figure 2: Flowchart of the WSN implementation

In addition to the altimeter, another POSIX thread will be created which will terminate the simulation. The user has the option to set a duration for the simulation, or end manually by entering -1. For our implementation, threads are terminated through shared flags whereas the nodes terminate after receiving a termination message. Once all the processes terminate successfully, the application ends.

## Base station

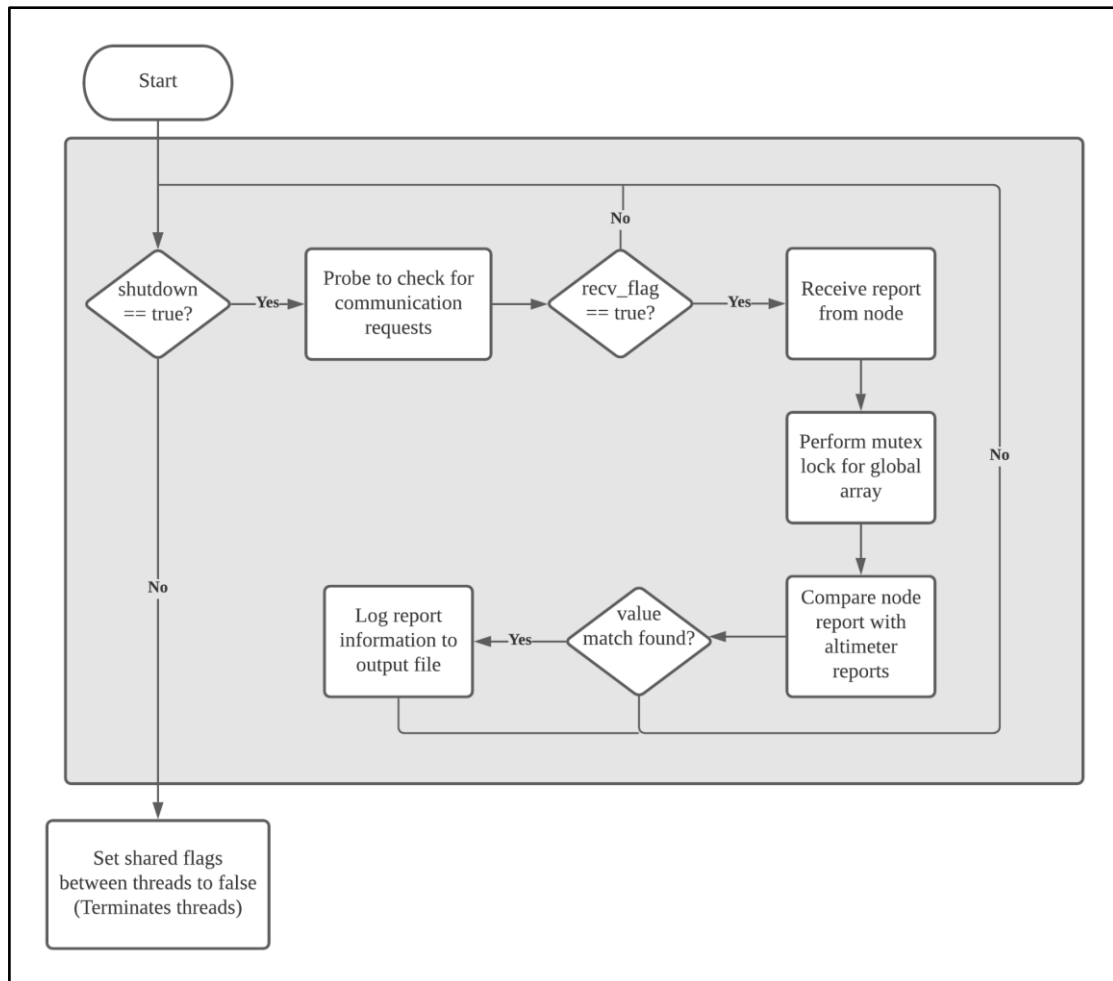


Figure 3: Flowchart of the base station simulation

The production of reports is carried out by the base station node whenever an alert report matches the altimeter reading. Whenever the base station receives a report from one of the sensor nodes via `MPI_Recv`, it will cross-check the related contents of the report with the contents which are produced and stored in the global array by the satellite altimeter. Then, whenever a match is found, the base station uses the contents of both the global array and the node report to generate a report which is then written to a text file which compiles the reports generated during runtime.

An additional thing to note is that whenever the global array is being accessed by the base station for cross-checking, the code performs mutex lock on the global array until it finishes checking the values in the array to address race conditions. Once it is done, the mutex is unlocked so that the altimeter can continue updating values in the global array.

## Sensor nodes

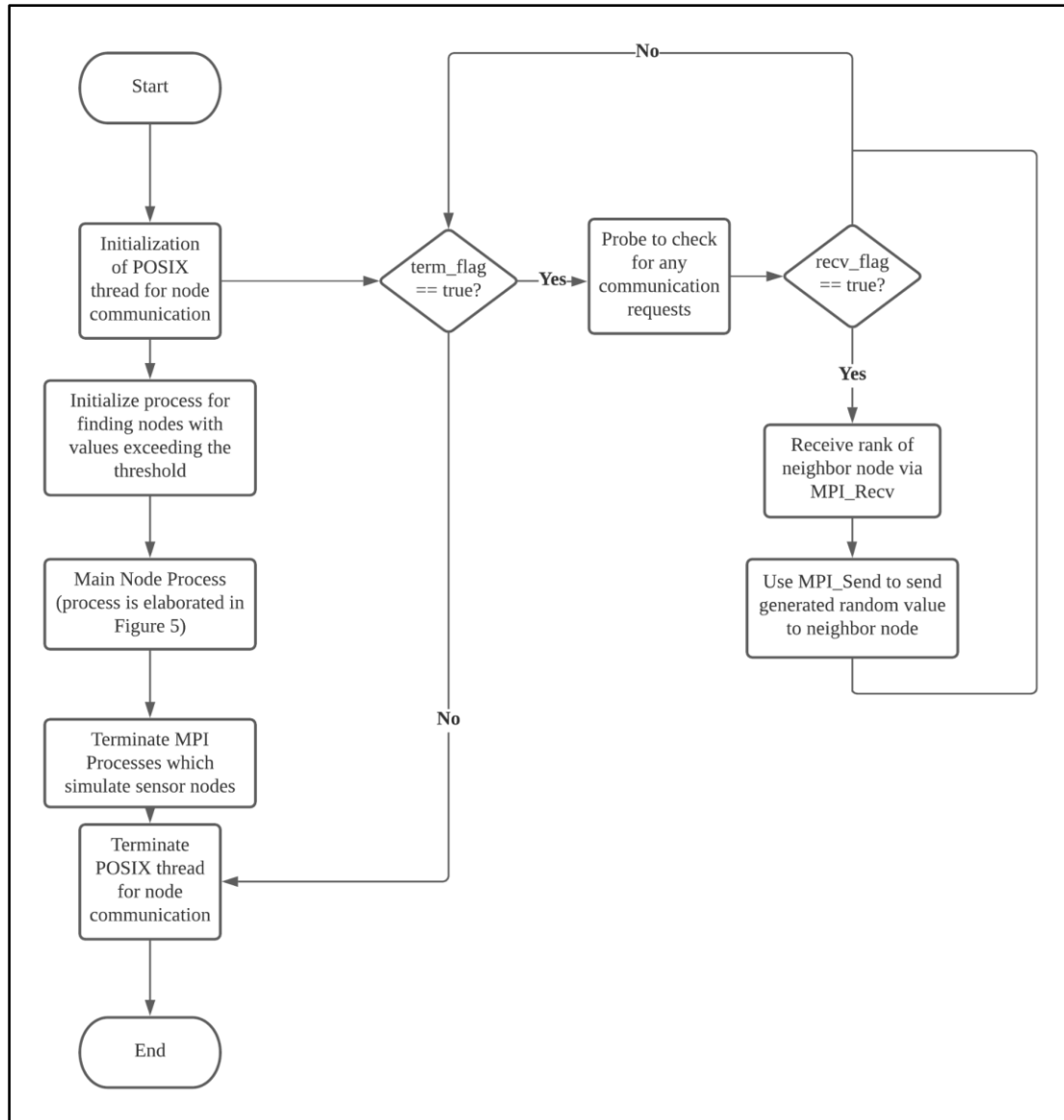


Figure 4: Flowchart of the sensor node simulation

As illustrated in Figure 4, each node will create a POSIX thread which will be responsible for handling value exchange between neighbour nodes. This is so all nodes can respond to these requests while performing its main process. Additionally, `MPI_Barrier()` is used to synchronize the processes which also prevents race conditions.

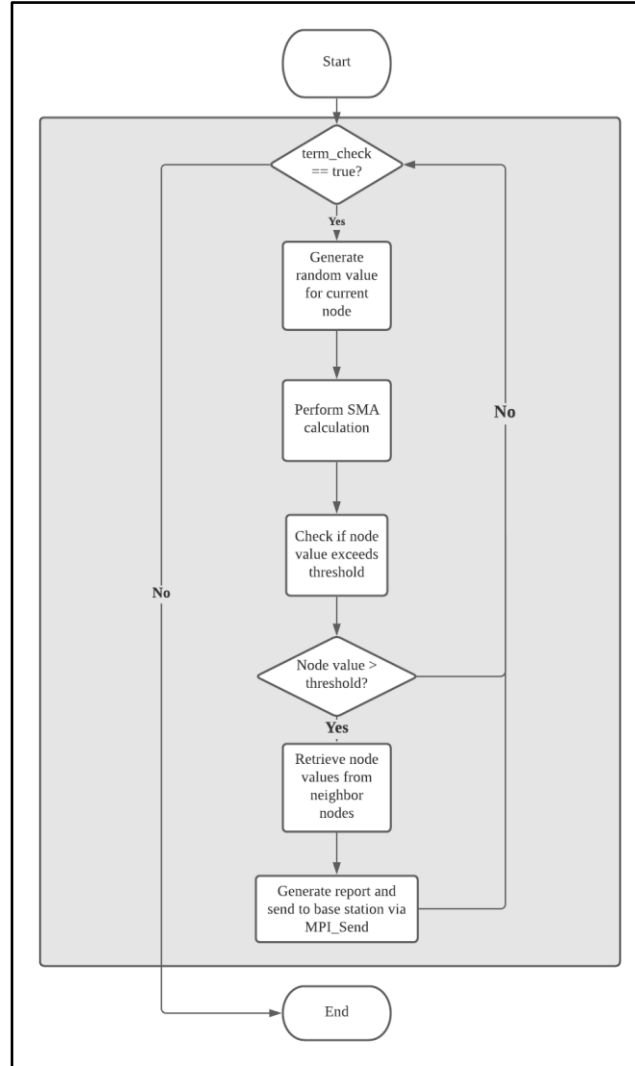


Figure 5: Flowchart of the sensor node's main thread

In every iteration, each node will generate a random value between a fixed range (5500, 7000) which will be used to calculate its simple moving average. Whenever a node is found to exceed the threshold value, a request will be sent to the adjacent nodes to retrieve the heights for those nodes as well. If the neighbour node's values match within a given tolerance range, the node begins to compile all the contents required for the report into a single struct, which is then sent to the base station node for cross-checking later on.

### Simple Moving Average

The simple moving average of the sensor nodes is based on Equation (1), which is the formula for calculating the moving average. For the calculation of the initial  $k$  values, the cumulative moving average formula will be used, which is shown in Equation (2).

$$SMA_{k,next} = SMA_{k,prev} + \frac{1}{k} (p_{n+1} - p_{n-k-1}) \quad (1)$$

$$CMA_{n+1} = CMA_n + \frac{x_{n+1} - CMA_n}{n+1} \quad (2)$$

For code implementation, a queue data structure is required for retrieving the previously calculated  $n$ -kth value. As such, previous  $k$  values are stored in an array that is maintained using a FIFO approach, similar to the altimeter's shared global array.

## Altimeter

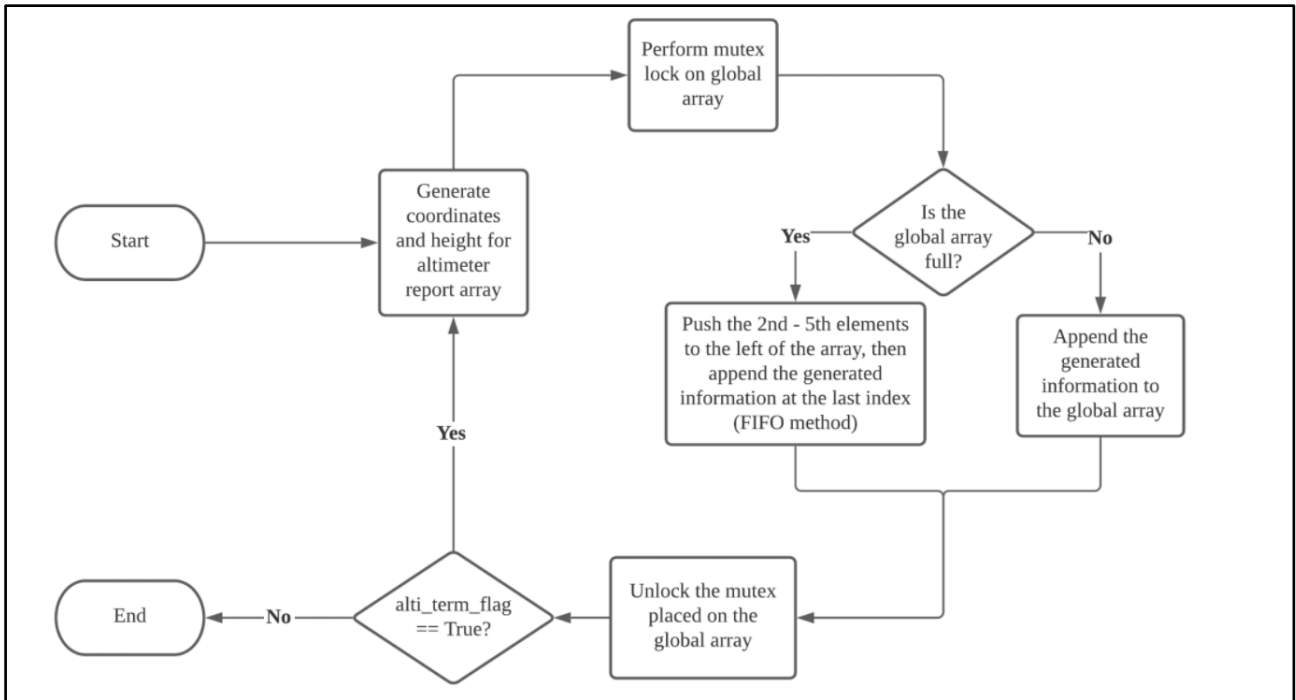


Figure 6: Flowchart of the altimeter's simulation

The diagram above illustrates the behaviour of the altimeter. To elaborate, the altimeter's behaviour is simulated by creating a POSIX thread which runs a function that generates a random coordinate with a random height that is over 6000 m. The values used for generating the random height are as followed:

- $x$  = random value
- $y$  = threshold value
- $z$  = max extra height

This random height is calculated through Equation (3) shown below.

$$\text{Height} = y + (x \bmod z) \quad (3)$$

Once the height value is generated, it is added to a global array of maximum size 8 via the First In First Out (FIFO) method. Whenever the global array that stores the information becomes full, the code pushes the 2nd to 8th elements to the left (which gets rid of the 1st element), and makes space for a new report to be added at the last index of the array.

Each time it is doing this, the code performs mutex lock to prevent other threads from accessing the global array. It only unlocks the mutex once the array is finished with its updates.

## B. Results Tabulation

### Sample Log

Our results will be based on the log reports produced by the system at the end of execution. These log reports will be included in the compressed zip file submitted alongside the report.

```
-----
Iteration: 3
Alert time: Mon Oct 18 01:58:35 2021
Alert type: Match

Reporting Node   Coord   Height (m)   IPv4
4               (1,1)   6297.000     172.16.227.105

Adjacent Nodes   Coord   Height (m)
1               (0,1)   6019.000
7               (2,1)   6251.334
3               (1,0)   6200.667
5               (1,2)   6648.000

Satelite altimeter reporting time: Mon Oct 18 01:58:33 2021
Satelite altimeter reporting height (m): 6129.829
Satelite altimeter reporting Coords: (1,1)

Communication Time (seconds): 0.00001023
Total Messages send between reporting node and base station: 1
Number of adjacent matches to reporing node: 3
Max. tolerance range between nodes readings (m): 300
Max. tolerance range between satelite altimeter and reporting nodes readings (m): 300
-----
```

Figure 7: Sample log that is written to the text file for reports

The system will output messages to the terminal to inform users of its current state. Figure 8 shows a sample of these messages.

```
-----
Duration parameter provided, base station will run for 60 seconds.
Time limit reached, base station will now begin shut down sequence
Altimeter terminated successfully
Termination message successfully sent to sensor nodes
Sensor nodes terminated successfully
Base station will now shut down
-----
```

Figure 8: Sample terminal output during code execution



## System specifications

Table 1: Details of parameters used and defined constants

Parameter	Value	Description
Width	3	Number of columns in the grid layout
Height	3	Number of rows in the grid layout
Duration	60	Duration which the system will run before termination
Threshold	6000	Threshold value
Defined constants	Value	Description
Interval	1	Time interval between iterations
Node Tolerance	300	Tolerance range for both nodes to be considered matching
Altimeter Tolerance	300	Tolerance range for node report and altimeter reading to be considered matching
Random max	7000	Upper bound for random height value generation
Random min	5500	Lower bound for random height value generation

Table 2: Specifications of local machine

Specification	Value
Processor(s)	4
Base Memory	8192

Table 3: Specifications of parameters used for running MonARch job script

Specification	Value	Description
ntasks	10	Number of tasks
cpus-per-task	3	CPU cores per task

## Summary of results

Table 4: Summary of results when running using local machine

Run	Reported events	Total communication time
1	106	0.276273
2	91	0.383401
3	111	0.370016
<b>Max</b>	111	0.383401
<b>Min</b>	91	0.276273
<b>Average</b>	102.667	0.34323

Table 5: Summary of results when running using MonARch

Run	Reported events	Total communication time
1	121	0.003009
2	121	0.003715
3	96	0.002705
<b>Max</b>	121	0.003715
<b>Min</b>	96	0.002705
<b>Average</b>	113	0.003143

## Line charts

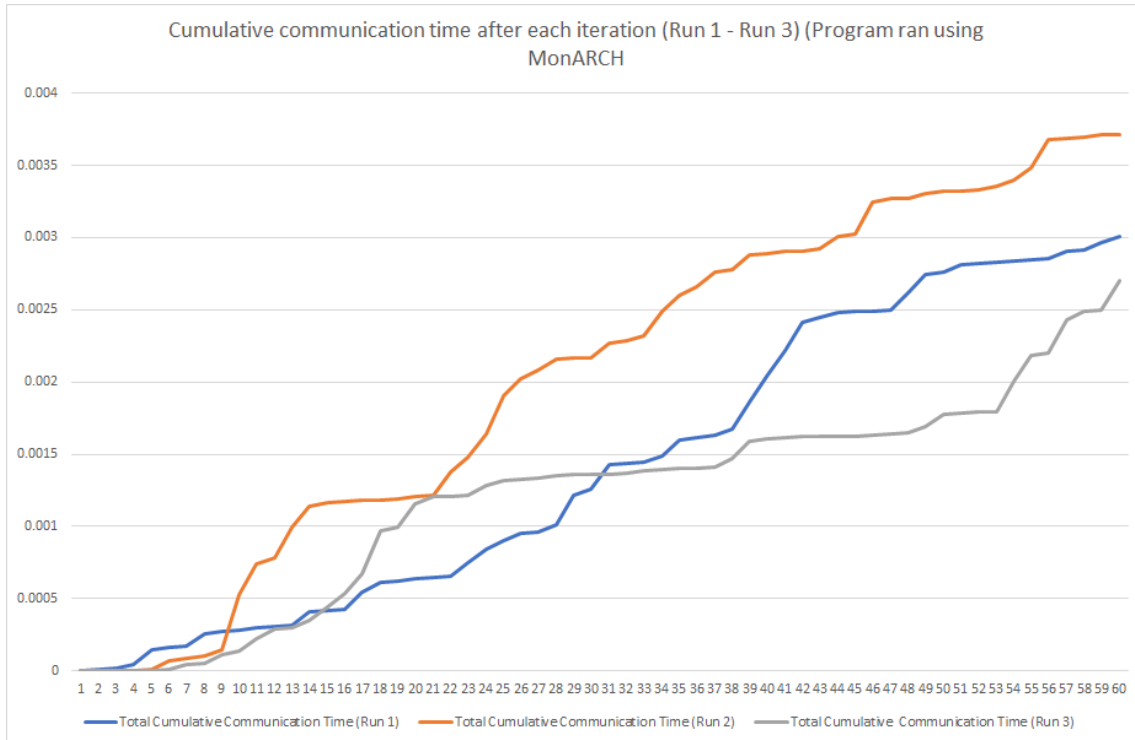


Figure 9: Cumulative communication time after each iteration (Run 1 - Run 3) (Program ran using MonARCh)

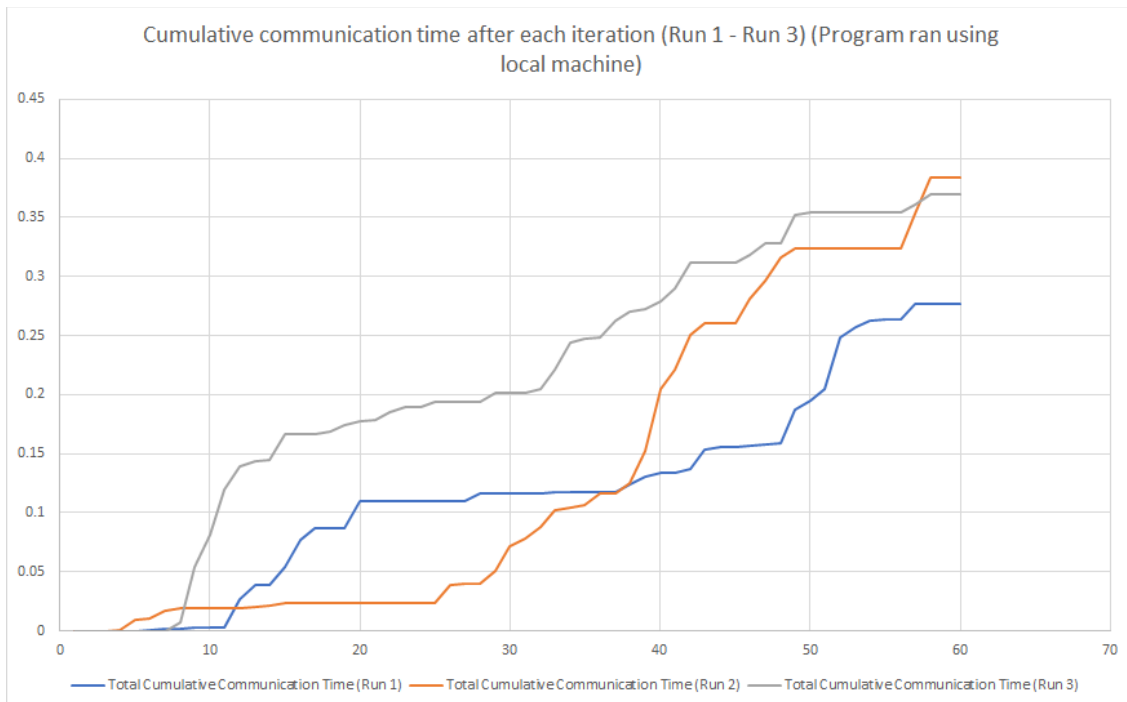


Figure 10: Cumulative communication time after each iteration (Run 1 - Run 3) (Program ran using local machine)

## Bar charts

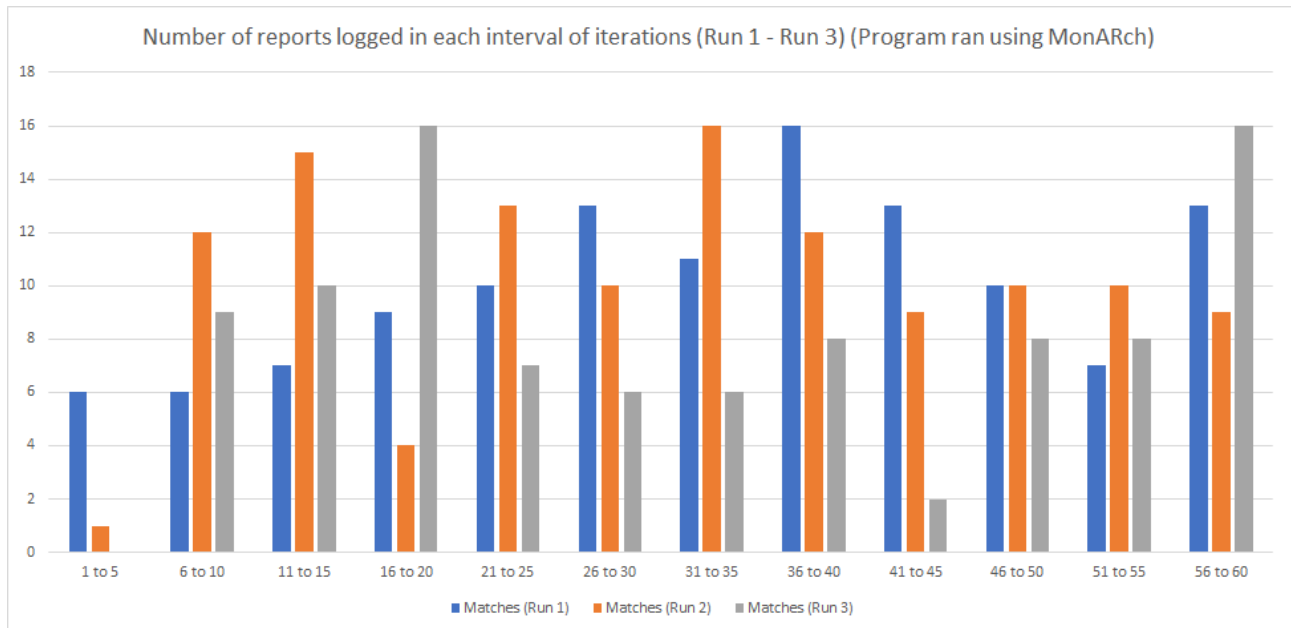


Figure 11: Number of reports logged in each interval of iterations (Run 1 - Run 3) (Program ran using MonARCh)

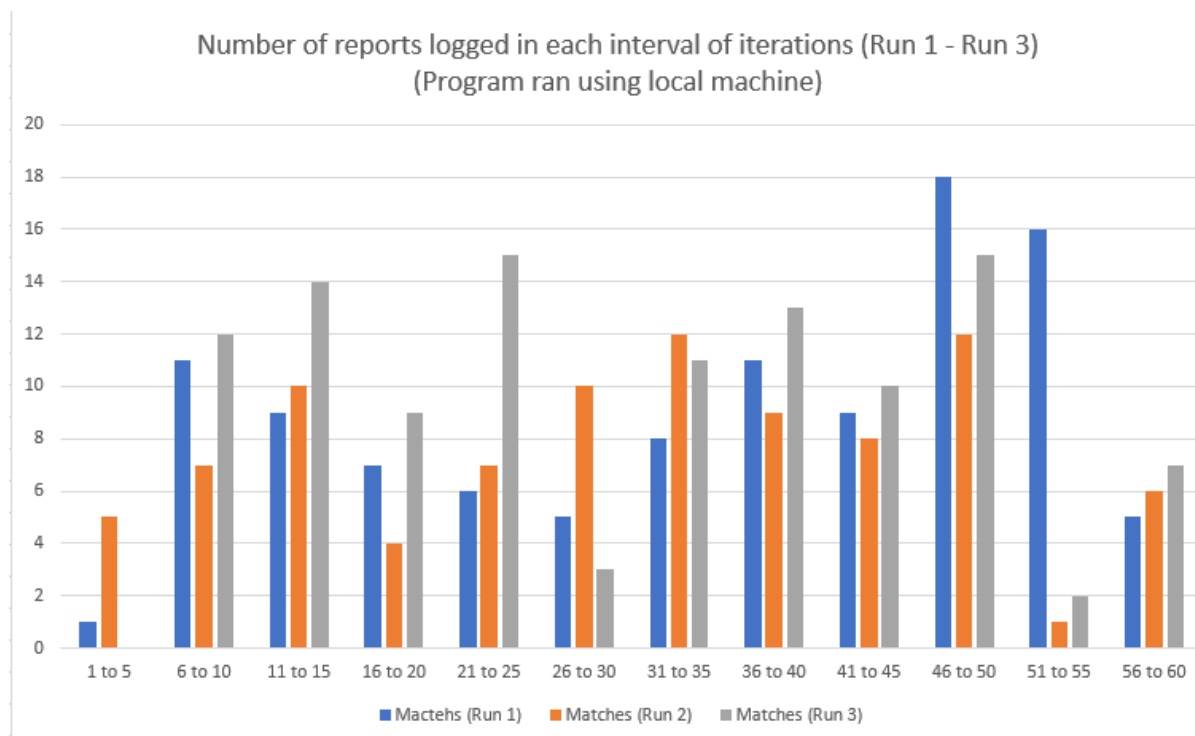


Figure 12: Number of reports logged in each interval of iterations (Run 1 - Run 3) (Program ran using local machine)

### **C. Analysis & Discussion**

We hypothesized that the average number of matches reported by the base station would increase until it reaches the 8th iteration, as the global array would be fully populated at that iteration. With this in mind, we proceeded to run the program with the parameters specified in Table 1. After retrieving the results of the program, we would then tabulate the results of the run. This process is then repeated 3 times so that we could more accurately compare and analyse the results. Once all the results have been tabulated, a grouped bar chart is created to analyse the results. The grouped bar chart in Figure 8 shows the results of the 3 runs of the program on MonARCH.

When analysing Figure 8, we can see that the grouped bar chart supports the hypothesis we stated earlier. Around iterations 1 to 5, we can see that there is a very low average number of reports being logged, with some runs having no reports to make at all. Once the iterations go past the 6 to 10 mark, we can see that the average number of reports being logged starts to become more stable.

An additional hypothesis we have is where the communication time will grow in a linear manner, while the probability for events to occur are the same for any iteration. However, based on the results shown in Figure 10, we can observe that the cumulative growth between intervals is inconsistent. If we analyze run 2 we can observe that the rate of increase spikes after the 30th iteration. We believe that a possible cause would be how the alerts are usually clustered when one node exceeds the threshold. To elaborate, when a node matches the neighbour node, there is a high possibility that the neighbour node may exceed the threshold as well. Due to this, the number of requests would increase greatly for a small part of the cartesian grid. As such, each node may require handling the request from all its neighbors. This causes communication to be close to a serialized process as our implementation has each node's thread handle at most one neighbour node's request at any given time. A similar issue may be possible on the base station as well as only the main thread handles reports from sensor nodes. A possible fix to mitigate this issue would be to increase the number of threads for handling communication requests between the nodes and base station.

In terms of analysing the communication time behaviour of the sensor nodes, we created line graphs to illustrate the cumulative communication time between nodes. These graphs can be referred to at Figure 8 and Figure 9. If we relate these line graphs to the grouped bar charts at Figure 10 and 11, we can see that the increase in communication time is proportional to the number of matches found at each iteration interval. This is because the more node values that are found to have exceeding thresholds, the more inter-node communication required before the report is sent to the base station, therefore increasing the cumulative communication greatly.

## **D. References**

Note: You may opt to customize this report to include additional sub-sections or any additional formatting where necessary.

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: \_\_\_\_\_ 

Signature of student 2: \_\_\_\_\_  \_\_\_\_\_

Signature of student 3: \_\_\_\_\_