MONASH University

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
30509386	Do	Hoang Khuong Duy

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

Unit name and code	Parallel Computing FIT3143	
Title of assignment	Assignment 2	
Lecturer/tutor	Paul Miller, Hoai Nguyen	
Tutorial day and time	9AM-11AM Thursday	**Campus** Clayton
Is this an authorised group assignment? **Yes** **No**		
Has any part of this assignment been previously submitted as part of another unit/course? **Yes** **No**		
Due Date 18/10/2021		Date submitted 16/10/2021

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.

MONASH University

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 The material used can be from any source (staff, students or the internet, published and unpublished works).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……16/10/2021……*

* delete (iii) if not applicable

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

FIT3143 Semester 2, 2021
Assignment 2 - Report

Team Name (or Number): _____17_____

Student email address	Student First Name	Student Last Name	Contribution %	Contribution details*
hdoo0010@student.monash.edu	Hoang Khuong Duy Do	Do	100	I finished both the coding part and the report part by myself. My partner doesn't respond to my message and is not making an effort to contribute.

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): 1947

A. Methodology
1502

B. Results Tabulation
239

C. Analysis & Discussion
206

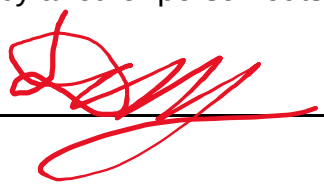
D. References

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

MONASH University

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:_____

Design and Implementation of a Tsunami Detection in a Distributed Wireless Sensor Network

Faculty of Information Technology

Monash University, Clayton

Melbourne, Australia

hdoo0010@student.monash.edu

Abstract- Wireless Sensor Networks (WSN) are used commonly in monitoring environmental habitats, it helps reduce cost and time for forecasting and resolving environmental issues. This report aims to apply and implement the WSN in reading and processing sea water level for tsunami detection using a 2D Cartesian grid topology of tsunami sensors, a satellite and a base station. The sensor nodes can communicate with their direct neighbour nodes to compare their water level readings and alert the base station if a certain threshold is exceeded. The base station will then obtain a reading from the altimeter satellite at that same time and compare them and log the result of the alert whether it is a false or real alongs with the vital information. To design this simulation, the Message Passing Interface (MPI) library in C is used for parallel communication across the network.

A/ Methodology

A tsunami represents a series of waves in a water body caused by the displacement of a large volume of water, generally in an ocean or a large lake. Their destructive power can be enormous and therefore various government and non-government organisations around the globe are working together to set up sea based wireless sensors to detect abnormal ocean wave heights (or sea water column height) which would indicate a tsunami. Such sensors are called tsunameters. The

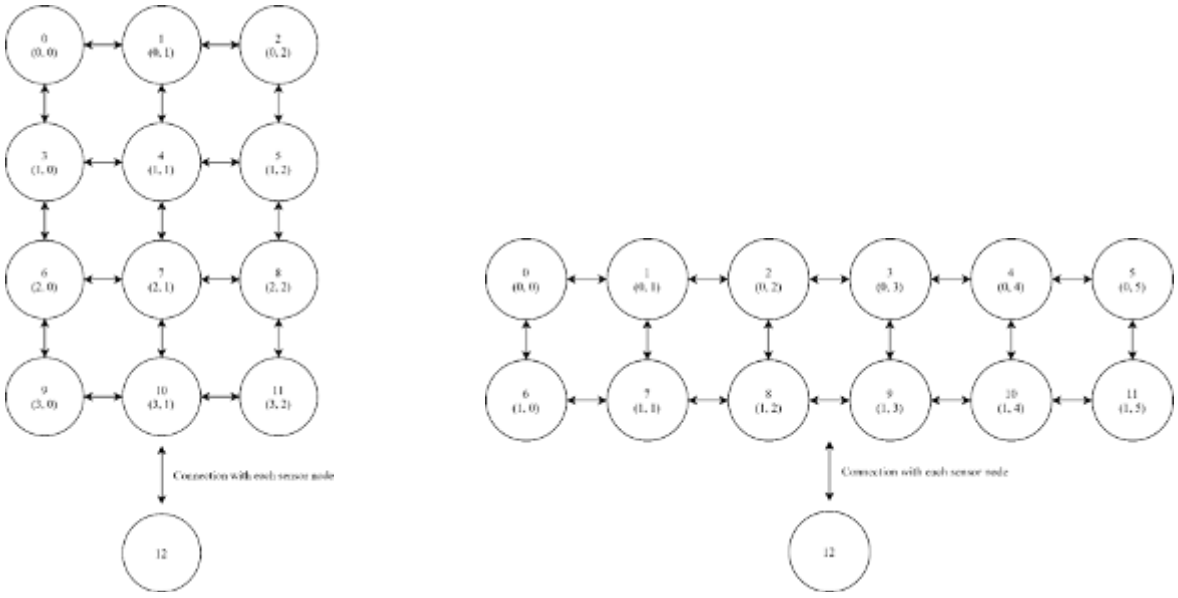
sea water column height readings from these sensors are then compared against similar readings from a satellite orbiting our planet. Should these readings be within a predefined threshold and time range, an alert is raised to the relevant authorities to evacuate coastal areas which could be affected by an oncoming tsunami.

This work attempts to simulate a wireless sensor network (WSN) to exchange data with a base station between processes. The motive for the implementation to use a 2D Cartesian grid topology is because this will offer scalability and efficiency.

With the use of parallel processing, the readings from the sensor network and the satellite are assumed to be in sync in terms of time hence the time of the alert will only be calculated by the time the base station finished comparing the readings. The implementation made use of the Message Passing Interface (MPI) library which provides the functionality to initialise the topology with Cartesian communicator, mapping and shifting. The communicator allows the sensor nodes to send and receive readings from their direct neighbours and the base station.

1/ Inter Process Communication Grid Architecture

The WSN uses a 2D Cartesian grid-base architecture where each node communicates with its direct adjacent nodes and the base station only. The number of nodes in the WSN is specified by the user at the start. The last rank is then selected as the base station while the rest as the sensor nodes. As the system also allows a dynamic grid arrangement, the user can also specify the corresponding row and column. Figure 1 illustrates how a grid can be rearranged for 13 nodes. Figure 1a demonstrates a grid where its row and column are 4 and 3 respectively, while Figure 1b demonstrates a grid where it is 2 and 6, respectively.



(a) 4 x 3 grid

(b) 2 x 6 grid

Fig. 1: Dynamic grid layout

The reason for the last rank to be the base station is because it is the only rank that receives user input to terminate the program through a sentinel value.

After initialization, each node can communicate with its adjacent four nodes (i.e. top, bottom, left and right nodes) and the base station. The base station will communicate with the sensor nodes to receive alert reports for a specified number of iterations. Multiple alert messages sent within the same iteration will be queued as the base station only handles one alert per iteration. To determine the coordinates of a given node in the Cartesian communicator group,

MPI_Cart_coords is used to compute the values. To put this into perspective, Equations 1 and 2 indicates the calculation of the row and column indexes of a given node. The base station will use the following equations to calculate the coordinates of the neighbours. This avoids the need of the sensor nodes to send the values over, thus reducing the size of the message to be sent.

$$row\_index = rank \,//\, columns \quad (1)$$
$$column\_index = rank \bmod columns \quad (2)$$

To separate the implementations between the base station and the sensor nodes, the communicator group needs to be split into two sets of processes in the form of a master/slave program. Algorithm 1 presents a pseudocode on splitting the processes where the last rank executes the base station function while the rest of the nodes execute the sensor node function.

```
/* Split communicator group into base station and sensor nodes. */
MPI_Comm_split(MPI_COMM_WORLD, my_rank == size - 1, 0, &new_comm);

if (my_rank == size - 1)
    base_station(MPI_COMM_WORLD, new_comm, iter, nrows, ncols);
else
    sensor_nodes(MPI_COMM_WORLD, new_comm, nrows, ncols, threshold);
```

The size of the communicator and the rank for each process is determined with MPI_Comm_size and MPI_Comm_rank respectively. After that, MPI_Comm_split is used to divide between the base station and the sensor nodes. The last rank will represent the base station, while the rest will be the sensor nodes.

After the split, a simple check is made to divide the functionalities. If the current rank is the last rank, the base station function is called. Otherwise, the sensor node function is called instead. Note that communication is still possible between the sensor nodes in the grid by using the new communicator, whilst they are still able to communicate with the base station through the original communicator.

2/ Sensor Nodes

Algorithm 2: Check if water height exceeds the given threshold and compare height readings between adjacent neighbours to determine if sending an alert report to the base station is necessary.

Each sensor node in the WSN will run Algorithm 2 periodically in an infinite loop until it receives an exit message from the base station. But if an exit message is not received yet, each node will periodically generate a height value at random.

After generating a water height reading for the sensor node's process itself, Algorithm 2 will attempt to obtain the height readings of the current sensor node's neighbour nodes using MPI_Isend, MPI_Irecv and MPI_Waitall.

A threshold of +200/-200 from the current node's height reading is set to verify that it is a result of tsunami. In addition, the number of neighbour nodes needed to be within the threshold needs to be 2 or more to identify as an alert. Once an alert is identified, each sensor node would build an array of information regarding the alert such as neighbour node's height readings, reporting node height reading and reporting node's rank. The array of information would then be reported to the base station using MPI_Send.

3/ Altimeter satellite

Algorithm 3 – A POSIX thread function that generates random height values that represent an water level height reading for each sensor node which would be appended into a buffer for comparison in the base station.

Algorithm 3 presents the function of a thread randomly generating the height readings to simulate the information obtained by the altimeter satellite which will be sent over to the base station.

The altimeter satellite's purpose being to provide height readings of the surface where the sensor nodes are located. These height readings will then be used for secondary comparisons with the reports received from the sensor nodes within a given time window.

Height values that are compared will need to fall within a threshold to identify the alert being a true alert or a false alert. Note that Algorithm 3 is called periodically as the base station listens to reports from the sensor nodes.

4/ Base Station

Algorithm 4 – The main process that receives any incoming messages from the sensor nodes and height readings from an altimeter satellite. It will compare the values between them and write/log to a file of each alert's details every iteration.

1) Initialization and Timer – After initialization in Algorithm 4 for the base station, we would use clock_gettime to start a timer for obtaining the total communication time between the base station and all the sensor nodes. The while loop is set to loop the amount of times that would be specified by the user to check for alerts.

2) Periodical Reports (Satellite and Sensor Nodes) – Every iteration, the base station would need to listen to incoming reports from the sensor nodes and obtain water level height values from the altimeter satellite periodically. pthread_create creates a POSIX thread which would run the function defined by Algorithm 3 to generate a random height for each process in the Cartesian topology. Later, pthread_join will be called to wait for the thread specified by tid to terminate then it will return immediately.

MPI_Iprobe is used as a nonblocking probe for incoming messages where it has the output parameter of flag. This serves as a message detector in our base station for any incoming messages from the sensor nodes. If a message is detected, the flag would be set to 1 by default which would trigger our while loop to process the alert or message.

3) Alert Processing – Once the flag is true, the base station would receive the message from the sensor node that sent the alert. Timestamps will be recorded such as the time of alert upon receiving the alert message from the sensor node and time of log to the text file. For each alert received from the sensor nodes, its height reading would be compared to the height reading from the altimeter satellite to identify false alerts. This introduces the idea where the hypothesis holds true where there might be more than 1 alert event sent from the nodes, but the base station is required to process the first alert event while the other alert events are queued in subsequent iterations. Hence, this causes an increase in communication time as there will be more queued alert events every iteration to run through where the height reading will not match the initial reading.

B/ Results Tabulation

A. Summary of implementation to obtain results

1) Base station listens to any incoming messages from the sensor nodes and incoming height readings from the satellite.

2) Base station compares height values of each alert with the water level height reading to determine if it is a true or false alert.

3) Base station generates the main log file which logs the alert event information and the entire simulation information.

4) Each sensor node gathers information of the detected alert and sends the information to the base station.

2) Summary of all Events – Figure 3 illustrates the last messages that would be logged into the log file by the base station.

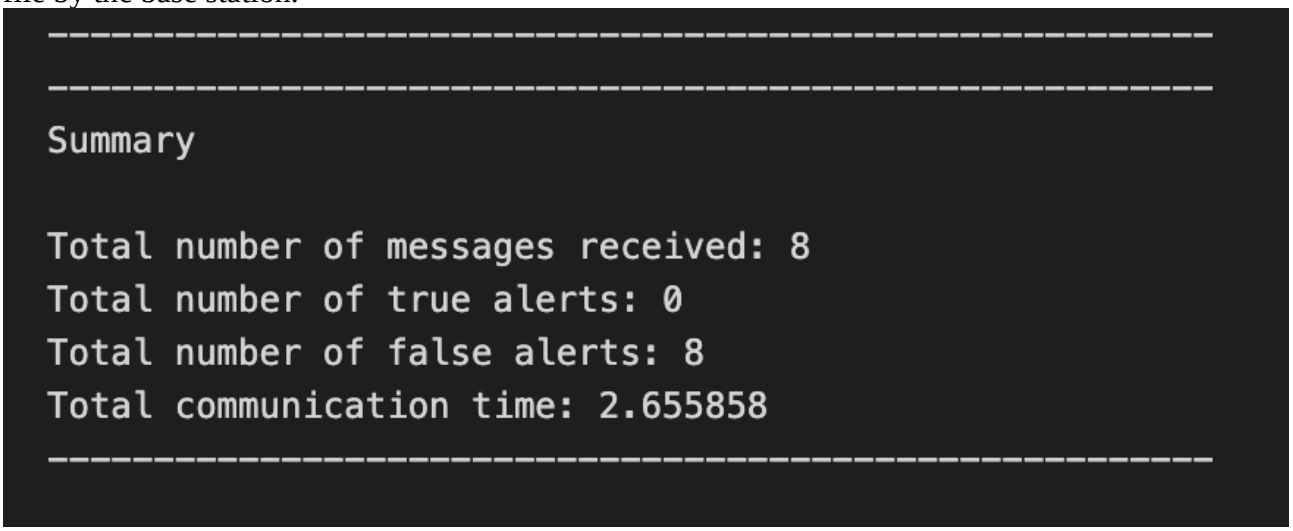


Figure 3

A) Event Summary after all Iterations – Total number of messages received represents the total number of alert events sent after each iteration. In addition, Total number of true alerts and Total number of false alerts are the information of the number of true and false messages received respectively after being verified with the altimeter satellite.

B) Communication Time – The communication time includes event detection from the sensor nodes after every iteration, receiving the message and height comparisons between the heights of the reporting node and the altimeter satellite. Hence, the communication time is calculated from the start of each iteration to the end of the iteration for the total length of communication between the base station and all sensor nodes for each iteration.

C/ Analysis and Discussion

In this report, we have demonstrated the usage of a 2D Cartesian topology using the MPI framework to model a simulated distributed wireless sensor network. By utilizing the properties of the MPI framework of IPC we were able to exchange data between nodes to simulate a WSN with foundational concepts of MPI. From the information gathered in the report, the finding of the hypothesis holds true as shown in Algorithm 4 of the base station that if there is an increase in simultaneous alert events reported by the sensor nodes in the Cartesian topology, it will result in a higher communication time as more alert events are queued up to be compared by the base station to the water level height readings.

We can come to a conclusion that even with the hypothesis that holds true, the grid based architecture in the form of a Cartesian topology has the benefit of faster event detection as there is direct link from the sensor nodes to the base station that communicates in parallel. This gives opportunity towards potential future work extending from this design in terms of further improvement in power efficiency of the sensor nodes to only report when necessary to result in a more efficient WSN.

D/ References

[1] Othman, M. F., & Shazali, K. (2012). Wireless Sensor Network Applications: A Study in Environment Monitoring System. Procedia Engineering, 41, 1204-1210. doi:10.1016/j.proeng.2012.07.302

[2] Open MPI: Open Source High Performance Computing. (2020). Retrieved 19 October 2020, from https://www.open-mpi.org/

[3] Amorim, J., Keizer, J., Miranda, A., & Monaghan, K. (2011). Forest fires research: Beyond burnt area statistics. Forest Science, 66(4), 415p411-415p419.

[4] What is Inter Process Communication (IPC)? - Definition from Techopedia. (n.d.). Retrieved October 21, 2020, from https://www.techopedia.com/definition/3818/inter-process-communication-ipc

[5] Hoefler, T., Rabenseifner, R., Ritzdorf, H., Supinski, B. R., Thakur, R., & Träff, J. L. (2010). The scalable process topology interface of MPI 2.2. Concurrency and Computation: Practice and Experience, 23(4), 293-310. doi:10.1002/cpe.1643