MONASH University
Information Technology

Design and implementation of a distributed wireless sensor network simulator to detect possible tsunami events

A. Methodology

The main function will split the communicators for the base station and the sensor nodes after validating the command line arguments.

Grid network concept:

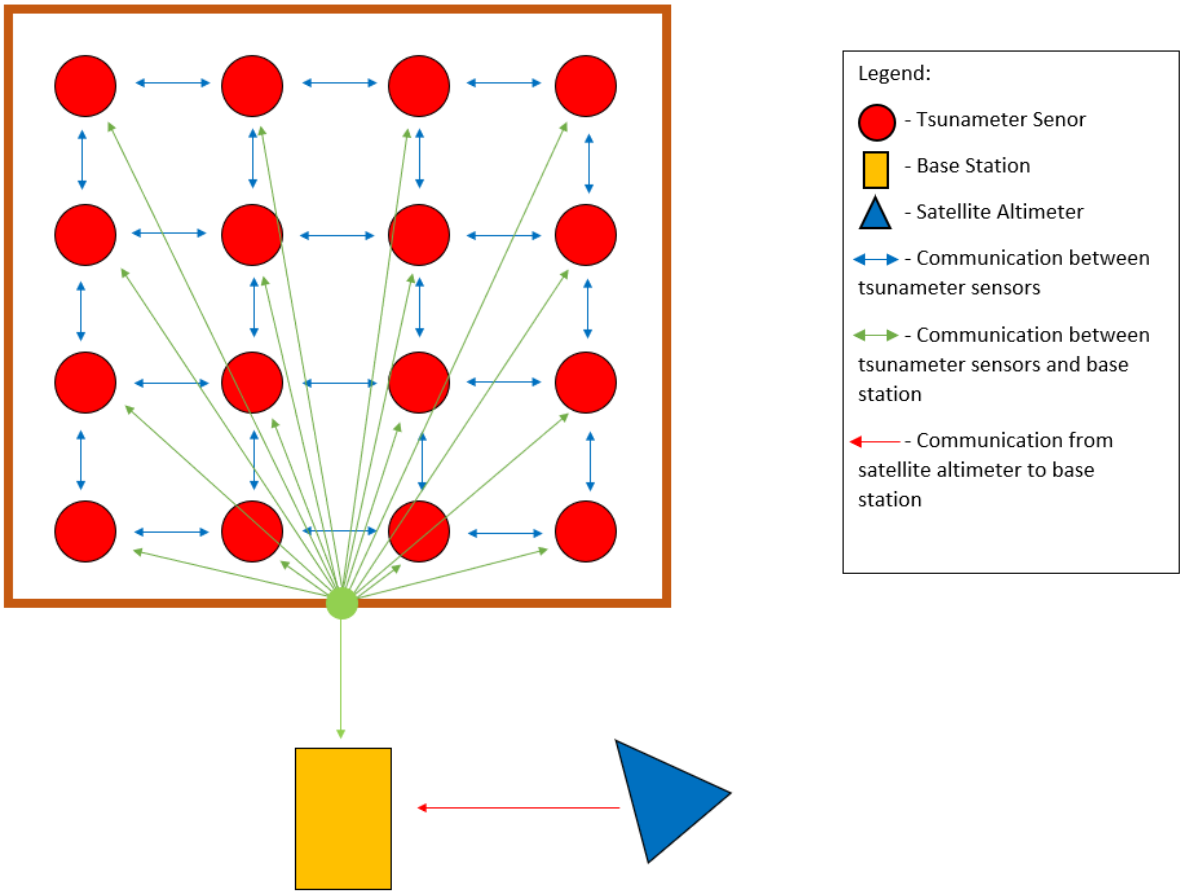


Fig 0. Grid Network

Our simulation network looks like the Figure shown above. The sensor nodes will be assigned to corresponding positions by creating the MPI Virtual topology. It will allow accessing the neighboring nodes in terms of such a matrix that is based on the user specified arguments (no. of processes) in an efficient and effective way without creating additional resources, and such a topology can be easily rescaled (Niethammer & Rabenseifner, 2019). The base station will be splitted as in a different communicator (which is just itself to distinguish from the sensor nodes), and perform the corresponding operations towards the sensor nodes. The satellite altimeter is just a thread inside the base station process itself, which is to independently continue generating simulation readings for the other thread to refer to any matches until it receives the termination signal.

MONASH University
Information Technology

Subsection A: Sensor Node

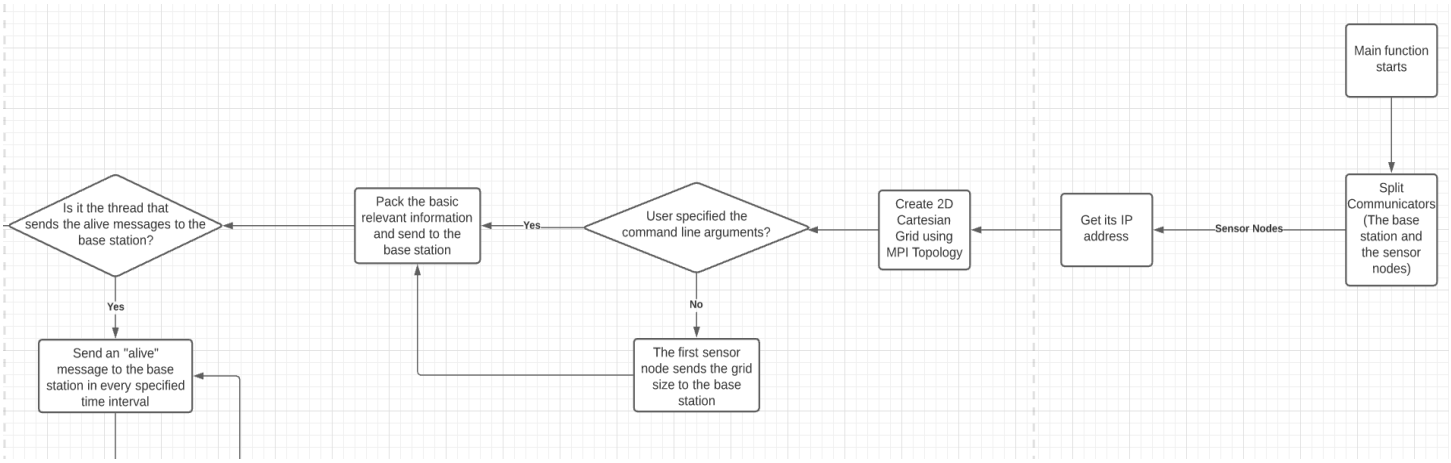


Fig 1. Pre steps of the sensor nodes

As shown above Figure 1 illustrates the pre-processing steps of the sensor nodes implementation. Other than initialising the relevant variables and getting the IP address of the current sensor node, the important part comes to the creation of the MPI 2D Topology, which is to assign the sensor nodes a different communicator that us better simulate the sensor nodes, that is the sensor nodes can easily reach its neighbors and coordinates by doing this. One interesting point to note is that if the user does not specify the command line arguments, which include the number of rows and columns for this virtual topology, the values will be initialised to 0 hence passing to the sensor nodes for further assignment, that is the function “MPI_Dims_create(size, ndims, dims)” will assign the values for number of rows and columns based on the number of processes provided. However since the base station also requires the number of rows and columns for some reasons, we will need to send the values after the new assignment by that function to the base station as otherwise the base station will have 0s for both, and to do this the sensor node with rank 0 will send the values after the creation. Basic information of the particular sensor node will be sent to the base station so that they won’t need to be sent again in the future.

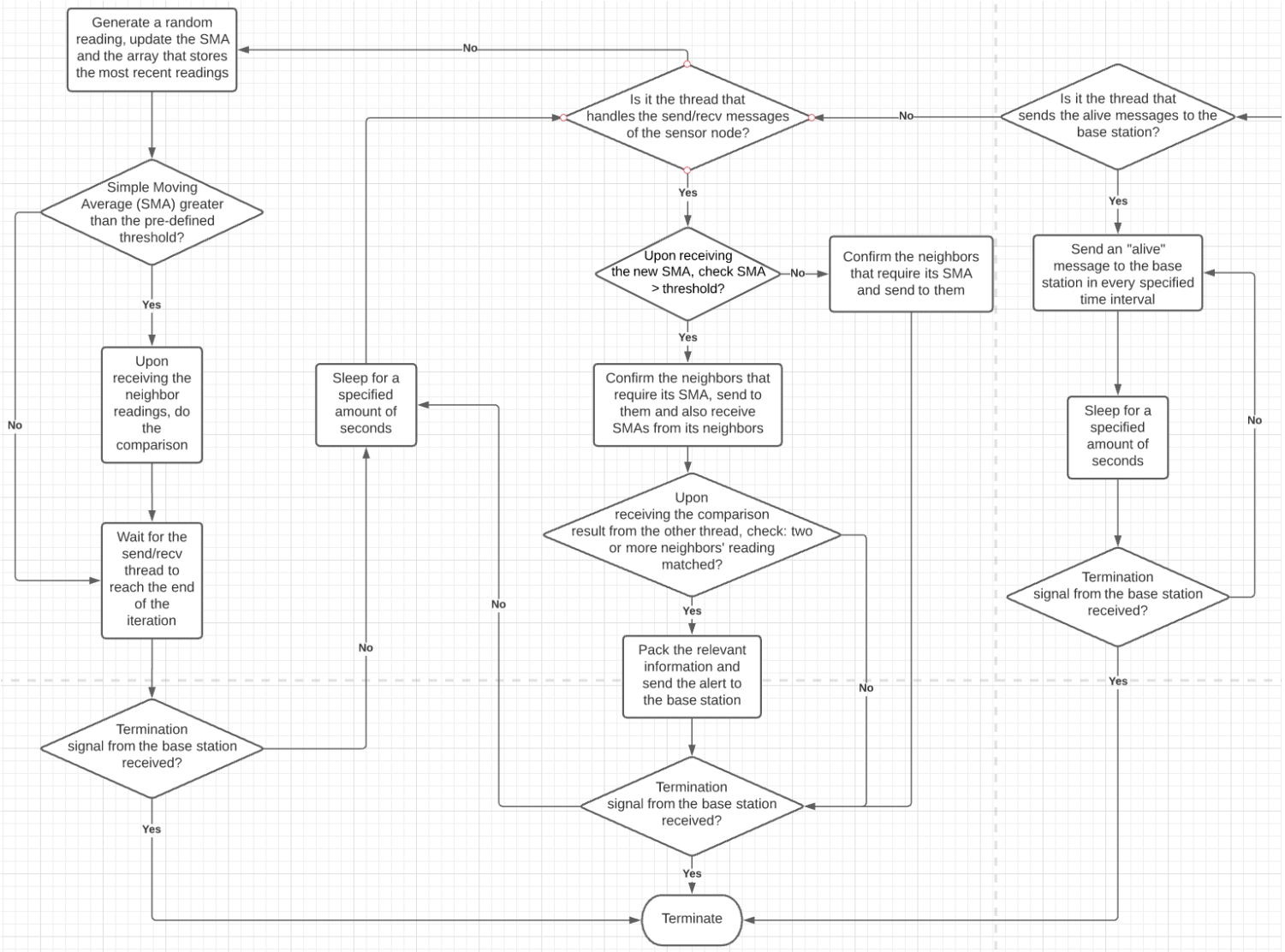


Fig 2. How the sensor nodes work

Figure 2 shows how the sensor nodes work after the virtual topology creation. Each sensor node has three different threads, one is in charge of sending “alive” messages to the base station, another has the responsibility to send and receive messages from the base station, and the final thread’s role of simulating a sensor reading, updating its SMA and requesting and comparing the readings with the ones from its neighbors.

The heartbeat thread uses MPI_Isend to send an alive message to the base station, to let it know that the node is still working. If this fails to send an “alive” message, this will be considered as a fault and will be detected and handled by the base station, that the fault will be logged and the base station will gracefully shut down the program. Such thread is independent of the others that it sends the message in its own timeline, but will terminate once the termination signal has been received by the other thread as to inline with the expectation.

The second thread that is to handle the send and receive operations will wait for the Simple Moving Average(SMA) from the last thread at the start of the iteration. Upon receiving the SMA, if the SMA is greater than the predefined threshold, it will confirm with its neighbors and send an alert to the base station if needed, otherwise it will just handle the messages from its neighbors. No matter whether its SMA is greater than the threshold, it will firstly tell its neighbors whether it needs the SMA so that its neighbors are aware of whether to send their SMA to that particular node. Upon confirming with its neighbors, it will send its SMA to the neighbors that require its SMA and post a set of receive operations to receive the neighborhood SMAs if its SMA is greater than the threshold. In that case this thread will need to wait for the last thread to compare the SMAs upon receiving and send an alert with all the relevant information (e.g., the time, SMA, etc...) to the base station if the result is that two or more neighborhood SMA readings matched. The thread will tell the base station whether it will send an alert at the appropriate time just so the base station can handle properly. The iteration will keep going until it receives a termination signal from the base station, and in that case both the other threads will terminate as well.

The last thread’s job is to calculate the new SMA based on the newly generated reading that is in the range of 5500

to 6500, and then compare its SMA with the ones from its neighbors upon receiving. This thread will wait until the send/recv thread reaches the end of its iteration so that the steps are consistent.

Subsection B: Base Station - Satellite Altimeter & Fault Detection Thread

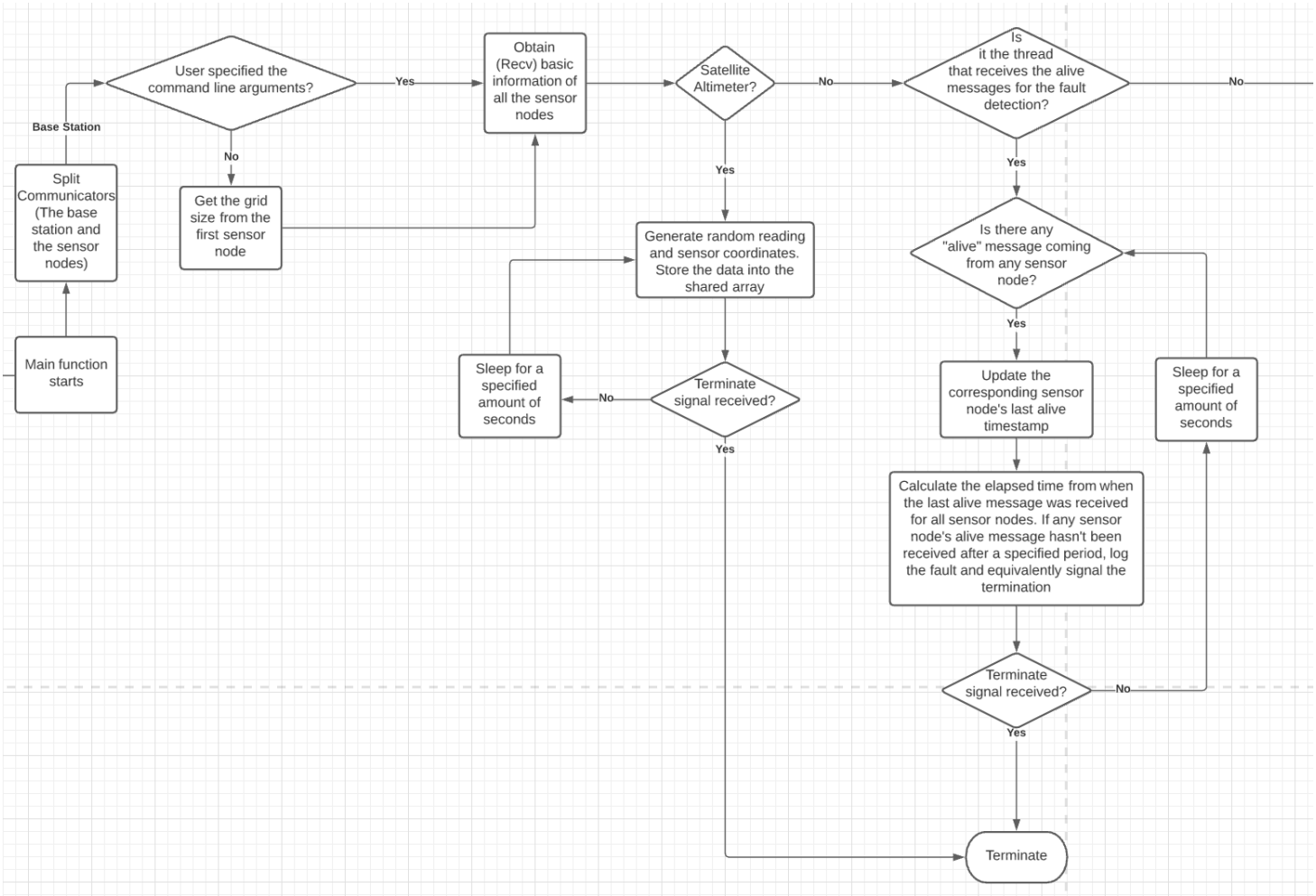


Fig 3. Pre steps of the base station, the Satellite Altimeter & the fault detection thread in the base station

In Figure 3, after the base station and sensor nodes have been split, the base station first checks whether the user has specified the sensor node grid size. If they haven't (i.e. the numbers of rows and columns are still 0), the base station instead receives the number of rows and columns from the first sensor node, as discussed above. The basic information of each node is then collected. This includes the nodes' coordinates, neighbours, IP address and their neighbours' IP addresses. One thing to note is that the relevant heap arrays for storing the information of the sensor nodes, the track of alerts and the simulation readings (by the satellite altimeter) will be initialised at the start of the base station and will be freed at the end as they are only used within the lifetime of the base station (Gamage & Baskaran, 2020).

The satellite altimeter does a relatively simple job that is to generate a random reading along with a valid pair of coordinates and then push the data into the shared global heap array (global in terms of the base station itself) using a FIFO approach. To push the data into the array, it will loop through the entire array, and "insert" the data into the array if there's an empty cell by checking if the year is the CURRENT_YEAR which is predefined. Otherwise FIFO approach will be used by setting the elements to the ones in the next index, and updating the last one. A critical lock will be used for both cases to avoid race conditions while the other thread is reading at the same index.

The fault detection thread initialises another heap array to store the timestamps of the last "alive" message received for all the sensor nodes. While the termination signal is not received, it will probe any "alive" messages from any of the sensor nodes, and update the timestamp upon receiving from a particular node. It will also check in each iteration whether the timestamp has exceeded a predefined time tolerance range (based on the number of processes) for all the nodes, and if there's any case detected, it will be considered as a fault and then signal the termination after logging the fault into the file, hence the program will terminate. Otherwise the iteration will continue after sleeping for a specified amount of seconds (For the Monarch version the sleep has been removed due to the limitation).

Subsection C: Base Station - Send/Recv & Data Processing Threads

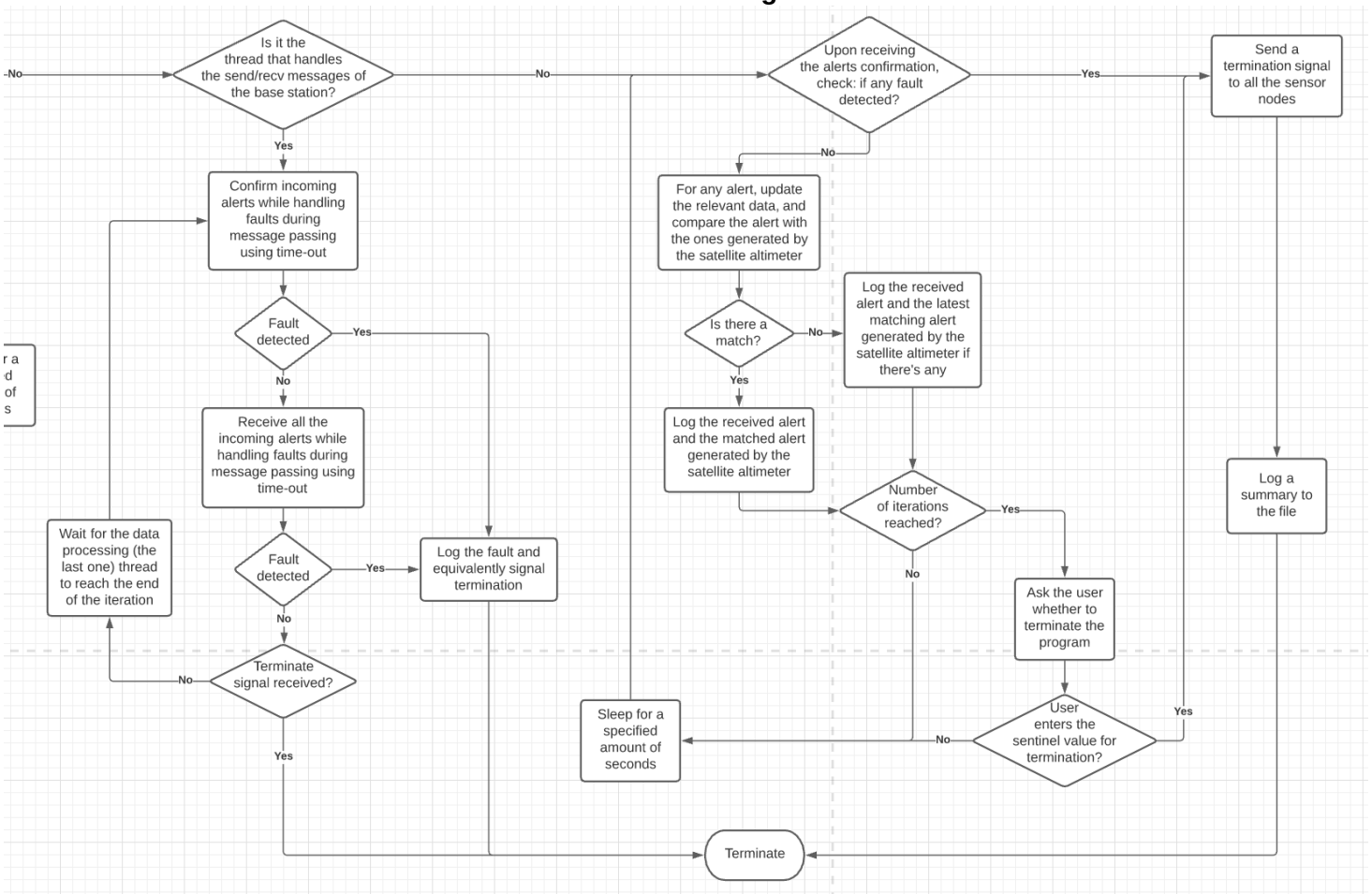


Fig 4. The Send/Recv thread and the data processing thread in the base station

In Figure 4, the next thread is in charge of all the send and receive messages from the sensor nodes. The first step is to check if any nodes are sending alerts to the base station in this iteration. Fault detection is also implemented here as well below when receiving the actual alerts, by using MPI_Test to keep checking whether the particular message has been received and only stops once it has been received or time out. If it's the latter case, termination will be signalled with logging the fault to the file, just like the fault detection thread. Otherwise the thread keeps receiving messages and stores them into an array until all have been collected. This thread will wait until the data processing thread has finished its job for consistency, otherwise it will terminate upon receiving the signal.

The final thread basically acts as the base station itself, for data processing. During each iteration, upon knowing the sensor nodes that are sending alerts in this iteration, without any fault detected, it will start processing the alerts upon receival, otherwise termination will be called with also signalling the sensor nodes. For any alert received without a fault, the information (time, SMA, etc...) will be unpacked and then compared against the ones in the shared array generated by the satellite altimeter with a critical lock to avoid race conditions (Data being changed during comparison). If there's a match within tolerances, the alert will be logged as a match into the file with the matched alert in the array, otherwise it will be logged as a mismatch and look for the latest alert for that sensor node in the shared global array and log it if there's any. The user will be prompted to decide whether to terminate the program, if not the iteration will be reset to initial. Termination status (0 = false, 1 = true) will be sent to the sensor nodes at the end of each iteration to make sure they shut down properly if true. A summary with relevant information during the program will be logged at the end, including the case it's terminated by fault(s).

MONASH University
Information Technology

B. Results Tabulation

We managed to test the simulation on both a local virtual machine and MonARCH.

Platform Specifications:

	Local Virtual Machine	MonARCH
No. of available logical processors	6	30 per node
System memory	10.6GB	32GB
CPU frequency	3.80GHz	2.50GHz (according to the website). / with 1 CPU per task
network bandwidth	N/A	50 Gbit/sec (Uses Mellanox ConnectX-4c cards, each card is capable of a max of 50)
IP address	127.0.1.1	172.16.226.182; 172.16.226.142; 172.16.226.53;

Specifications of test runs:

	TEST A	TEST B	TEST C	TEST D
Number of runs	2	2	2	2
Grid Size (Rows x Cols)	4 x 3	5 x 4	5 x 5	7 x 5
Sea water column height threshold	6000	6000	6000	6000
No. of iterations at the base station	20	20	20	20

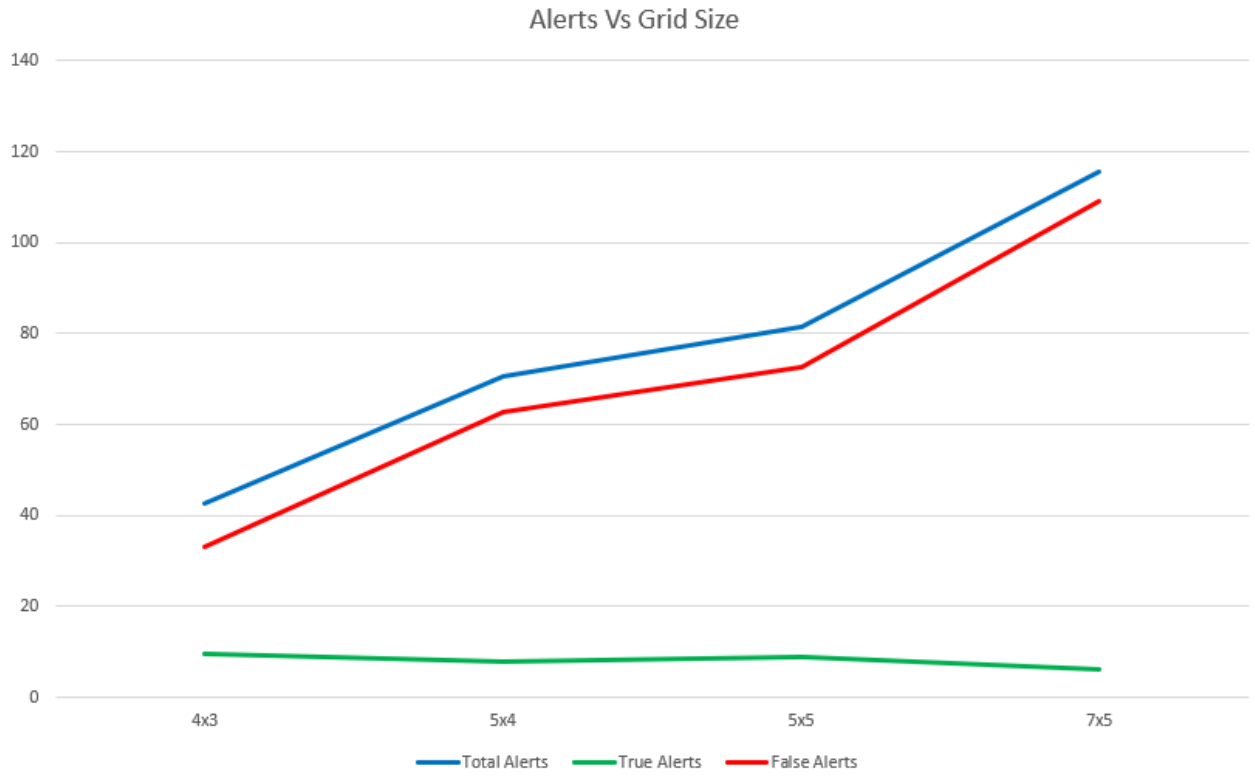
Test Results:

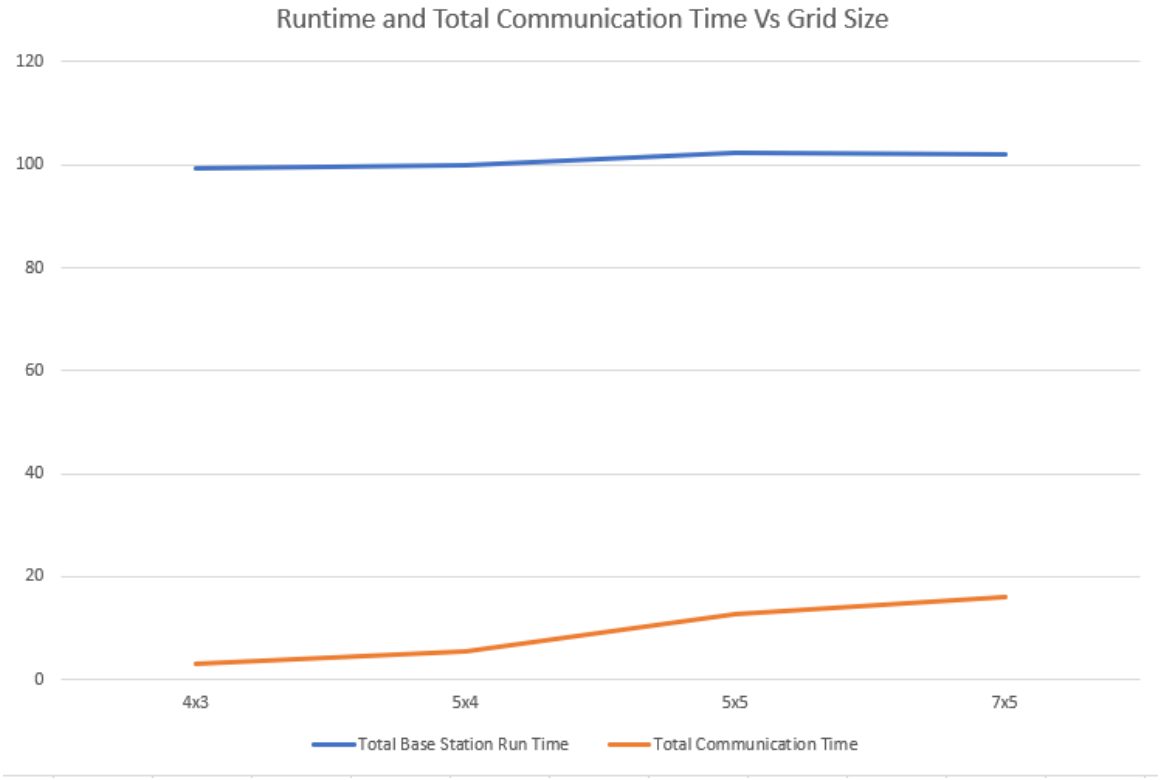
	TEST A	TEST B	TEST C	TEST D (Additional)
No. of Total alerts	----------Local---------- First run: 42 Second run: 37 -------MonARCH------ First run: 39 Second run: 53	----------Local---------- First run: 84 Second run: 85 -------MonARCH------ First run: 35 Second run: 78	----------Local---------- First run: 68 Second run: 114 -------MonARCH------ First run: 82 Second run: 62	----------Local---------- First run: 131 Second run: 106 -------MonARCH------ First run: 121 Second run: 104
No. of TRUE alerts	----------Local---------- First run: 8 Second run: 6 -------MonARCH------ First run: 13 Second run: 11	----------Local---------- First run: 11 Second run: 12 -------MonARCH------ First run: 3 Second run: 5	----------Local---------- First run: 8 Second run: 16 -------MonARCH------ First run: 5 Second run: 7	----------Local---------- First run: 5 Second run: 6 -------MonARCH------ First run: 5 Second run: 9
No. of FALSE alerts	----------Local---------- First run: 34 Second run: 31	----------Local---------- First run: 73 Second run: 73	----------Local---------- First run: 60 Second run: 98	----------Local---------- First run: 126 Second run: 100

MONASH University
Information Technology

	-------MonARCH------ First run: 26 Second run: 42	-------MonARCH------ First run: 32 Second run: 73	-------MonARCH------ First run: 77 Second run: 55	-------MonARCH------ First run: 116 Second run: 95
Total Base Station runtime (seconds)	----------Local---------- First run: 99.344 Second run: 99.684 -------MonARCH------ First run: 98.565 Second run: 99.330	----------Local---------- First run: 99.837 Second run: 100.297 -------MonARCH------ First run: 99.535 Second run: 99.445	----------Local---------- First run: 104.808 Second run: 104.35 -------MonARCH------ First run: 99.620 Second run: 101.053	----------Local---------- First run: 102.949 Second run: 103.832 -------MonARCH------ First run: 100.490 Second run: 100.284
Total Comm. Time (seconds)	----------Local---------- First run: 2.400 Second run: 2.162 -------MonARCH------ First run: 2.817 Second run: 5.108	----------Local---------- First run: 5.091 Second run: 4.980 -------MonARCH------ First run: 3.636 Second run: 8.592	----------Local---------- First run: 7.778 Second run: 15.79 -------MonARCH------ First run: 9.621 Second run: 17.006	----------Local---------- First run: 17.236 Second run: 14.530 -------MonARCH------ First run: - N/A due to multi-nodes Second run: - N/A due to multi-nodes
Average Comm. Time (seconds)	----------Local---------- First run: 0.057 Second run: 0.058 -------MonARCH------ First run: 0.072 Second run: 0.096	----------Local---------- First run: 0.061 Second run: 0.059 -------MonARCH------ First run: 0.104 Second run: 0.110	----------Local---------- First run: 0.114 Second run: 0.139 -------MonARCH------ First run: 0.117 Second run: 0.274	----------Local---------- First run: 0.132 Second run: 0.137 -------MonARCH------ First run: - N/A due to multi-nodes Second run: - N/A due to multi-nodes
Fault detected	NONE	NONE	NONE	NONE

Graph (Performance of the sensor nodes for an increasing grid size**):**





MONASH University
Information Technology

Screenshots:

a. **Log files:**

From VM:

```
--------------------------------------------------------------------
Iteration: 17
Logged time: 2021-10-16 22:57:35
Alert reported time: 2021-10-16 22:57:35
Alert type: Mismatch

Reporting Node      Coord       Height(m)   IPv4
4                   (1,1)       6070.303    127.0.1.1

Adjacent Nodes      Coord       Height (m)  IPv4
1                   (0,1)       6105.175    127.0.1.1
7                   (2,1)       5761.198    127.0.1.1
3                   (1,0)       5994.360    127.0.1.1
5                   (1,2)       5773.126    127.0.1.1

Satellite altimeter reporting time: 2021-10-16 22:56:42
Satellite altimeter reporting height (m): 6465.816
Satellite altimeter reporting Coord : (1,1)

Communication Time (seconds): 0.00530
Total Messages send between reporting node and base station: 6
Number of adjacent matches to reporting node: 2
Max. tolerance range between nodes readings (m): 100
Max. tolerance range between satellite altimeter and reporting node readings (m): 100
Time tolerance range between satellite altimeter and reporting node (seconds): 15
--------------------------------------------------------------------
```

From MonARCH:

```
 GNU nano 2.3.1                          File: WSN_Log_File.txt

******************************************************
Base Station Log File:
Number of sensor nodes: 12. Grid Dimension = [4 x 3]
Threshold: 6000
Number of Base Station Iterations: 20
******************************************************
--------------------------------------------------------------------
Iteration: 2
Logged time: 2021-10-16 23:26:48
Alert reported time: 2021-10-16 23:26:48
Alert type: Match

Reporting Node          Coord           Height(m)       IPv4
4                               (1,1)           6018.367        172.16.227.43

Adjacent Nodes          Coord           Height (m)      IPv4
1                               (0,1)           5764.802        172.16.227.43
7                               (2,1)           5999.841        172.16.227.43
3                               (1,0)           6011.148        172.16.227.43
5                               (1,2)           5855.175        172.16.227.43

Satellite altimeter reporting time: 2021-10-16 23:26:44
Satellite altimeter reporting height (m): 6114.986
Satellite altimeter reporting Coord : (1,1)

Communication Time (seconds): 0.05586
Total Messages send between reporting node and base station: 1
Number of adjacent matches to reporting node: 2
Max. tolerance range between nodes readings (m): 100
Max. tolerance range between satellite altimeter and reporting node readings (m): 100
Time tolerance range between satellite altimeter and reporting node (seconds): 15
--------------------------------------------------------------------
--------------------------------------------------------------------
Iteration: 4
                                                [ Read 986 lines ]
^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U
```

b. **A summary of events** generated for one of the runs in each test:
From Local VM:

```
**********Summary**********
Termination Result: Terminated normally

Time taken for the base station (seconds): 131.339
Total communication time (seconds): 1.958
Average communication time (seconds): 0.058

Total number of alerts      Number of TRUE alerts       Number of FALSE alertS
34                          4                           30
**********
```

From MonARCH:

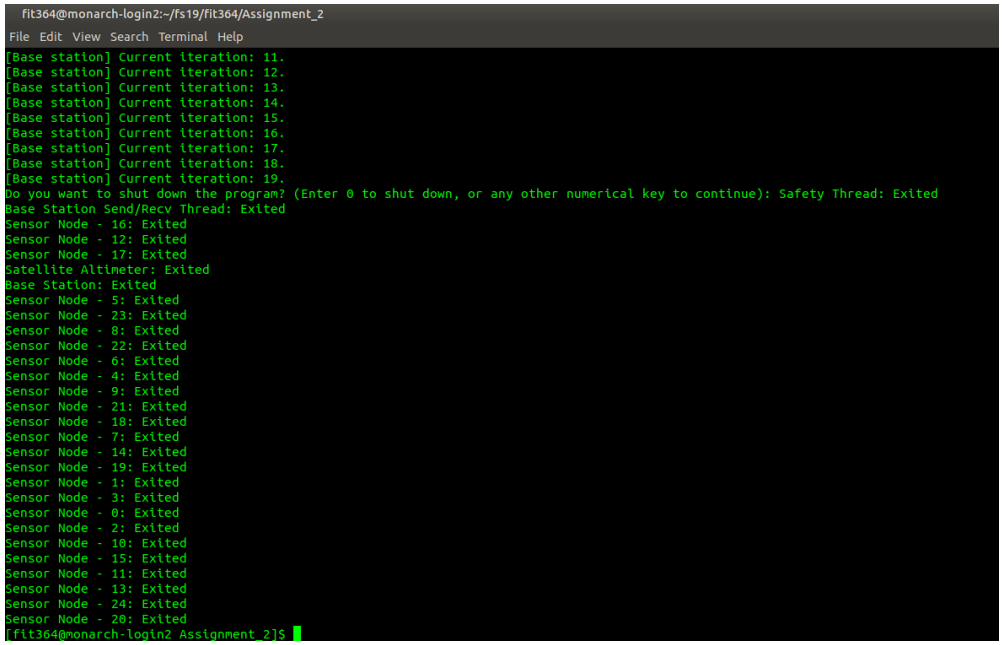
```
fit364@monarch-login2:~/fs19/fit364/Assignment_2
File  Edit  View  Search  Terminal  Help
 GNU nano 2.3.1                File: WSN_Log_File.txt

Communication Time (seconds): 0.06799
Total Messages send between reporting node and base station: 3
Number of adjacent matches to reporting node: 2
Max. tolerance range between nodes readings (m): 100
Max. tolerance range between satellite altimeter and reporting node readings (m): 100
Time tolerance range between satellite altimeter and reporting node (seconds): 15
--------------------------------------------------------------------------

**********Summary**********
Termination Result: Terminated normally

Time taken for the base station (seconds): 99.535
Total communication time (seconds): 3.636
Average communication time (seconds): 0.104

Total number of alerts          Number of TRUE alerts           Number of FALSE alertS
35                              3                               32
**********

^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Po
^X Exit          ^J Justify       ^W Where Is      ^V Next Page     ^U UnCut Text    ^T To Spe
```

c. **A Proper termination**:
From Local VM:

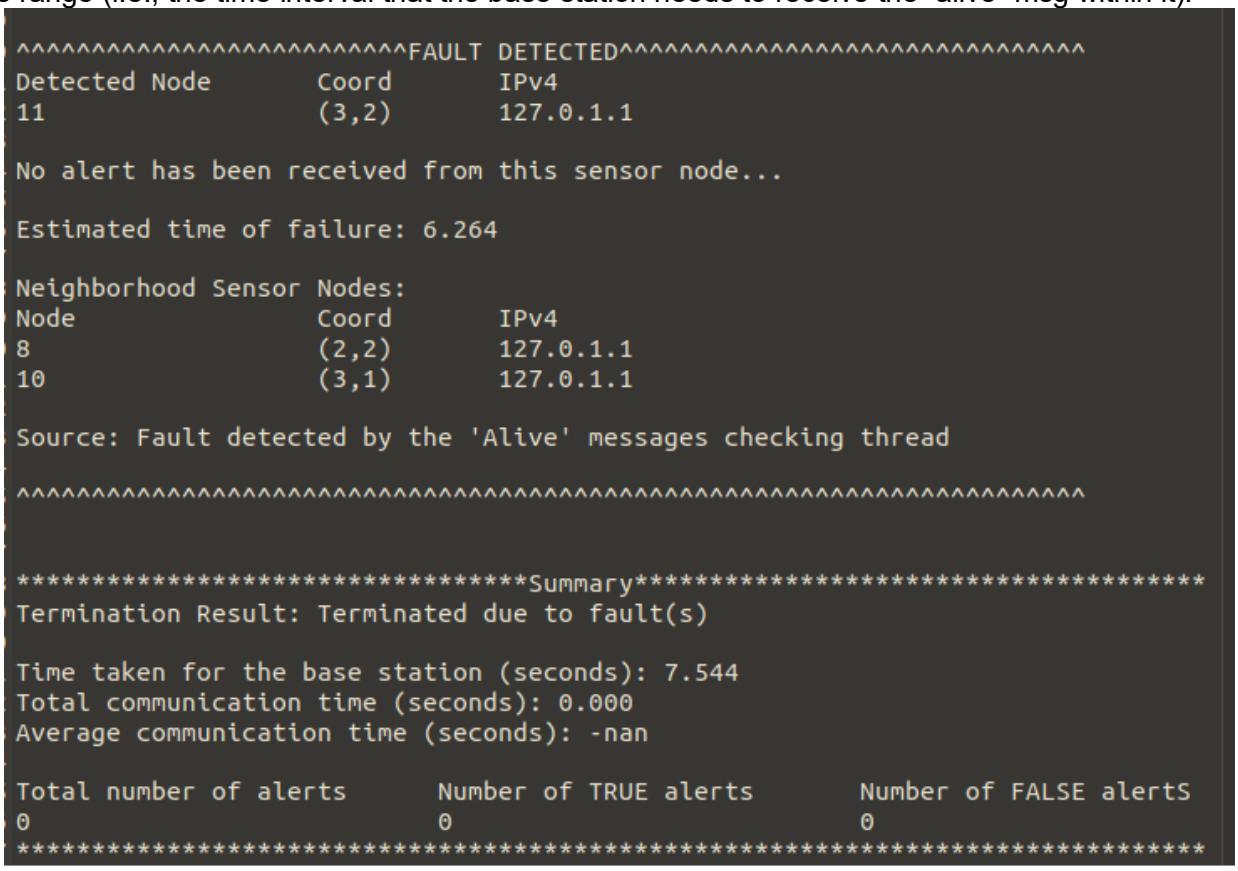
```
[Base station] Current iteration: 19.
Do you want to shut down the program? (Enter 0 to shut down, or any other numerical key to continue): 0
Base Station Send/Recv Thread: Exited
Sensor Node - 4: Exited
Sensor Node - 7: Exited
Sensor Node - 14: Exited
Sensor Node - 15: Exited
Sensor Node - 19: Exited
Sensor Node - 2: Exited
Sensor Node - 18: Exited
Sensor Node - 12: Exited
Sensor Node - 17: Exited
Safety Thread: Exited
Sensor Node - 5: Exited
Sensor Node - 6: Exited
Sensor Node - 9: Exited
Sensor Node - 10: Exited
Sensor Node - 1: Exited
Sensor Node - 8: Exited
Sensor Node - 13: Exited
Sensor Node - 11: Exited
Sensor Node - 3: Exited
Sensor Node - 16: Exited
Sensor Node - 0: Exited
Satellite Altimeter: Exited
Base Station: Exited
fit3143-student@fit3143:~/Desktop/FIT3143_Labs/Assignment_2$
```

MONASH University
Information Technology

From MonARCH:



Additionally we tested the **fault detection mechanism and the simulated response results** by lowering the time tolerance range (i.e., the time interval that the base station needs to receive the “alive” msg within it):



We are able to tell the source of the fault(s), how the fault(s) are detected and the other basic relevant information. The summary also states whether the program is terminated normally or by fault(s).

MONASH University
Information Technology

C. Analysis & Discussion - 730

Hypothesis: The amount of communication with the base station will be most dense in the middle of the grid due to an increased number of neighbors and communication between the nodes leading to a higher chance of successful event detection

The main, and most common ways in which the base station is communicated with is through the sensor nodes sending heartbeat messages, and sensor nodes sending alert messages. Since every sensor node is sending heartbeat messages every iteration, we can ignore these, as this will not determine which nodes are communicating with the base station more than others. Instead, only the alerts can be considered. As there are more neighbours in the centre nodes, it would logically follow that these would send more messages to the base station. This is because each node has about a 50% chance of generating a reading higher than the threshold, prompting a response from its neighbours. Each neighbour also has about a 50% to return a reading greater than the threshold, and only two are required to send an alert to the base station. The probability for a node to communicate with the base station is based on the number of surrounding nodes. These probabilities are:

2 surrounding nodes (corner): $1*(0.5^3) = 0.125$
3 surrounding nodes (edge): $4*(0.5^4) = 0.25$
4 surrounding nodes (middle): $11*(0.5^5) = 0.344$

Therefore the nodes in the middle are more likely to alert the base station than those on the outside. Additionally, the corner nodes are the least likely to communicate with the base station. This hypothesis and theory can be helped be proven with theoretical results, by counting the number of alerts received from the base station from each sensor node. The results after 60 iterations of a 4x3 grid are shown below:

13	15	12	7
16	22	15	13
7	13	10	5

Based on this, we can see the edges have three of the four least amount of alerts, the middle nodes have the highest amount of alerts, and the edge nodes are somewhere in the middle. It is worth noting that these results are based on probabilities, and while node (2,1) was less than node (0,1), this is likely due to chance. Therefore, based on statistical theory, with the aid of theoretical results, it can be concluded that communication with the base station is the most dense in the middle of the grid due to an increased number of neighbours.

From the results generated in the last section, we can see that the average communication time is generally below or around 0.1 second, intuitively the total communication time increases as the number of alerts increases. Basically the number of alerts increases as the grid size increases, however due to the randomness, a smaller number of alerts can also happen for a larger grid size. That being said, however it seems like the number of true alerts isn't affected by the number of alerts that much, there can be a reason that the number of simulation readings generated by the satellite altimeter is not changed, and the randomness plays a big role in terms of this as well. There isn't too much difference between the base station runtime of each grid size tested, this is mainly due to the same fixed number of iterations. The results from MonARCH (including multiple nodes) seem very similar to the ones generated using the local VM, despite it being less stable and there might be noticeable differences of the tiny timestamps between each run due the nature of the server. Although there could be slightly better performance (take less time to complete the runs) on MonARCH when multiple CPU and nodes are used.

There are, however, still a number of improvements we can make based on the tabulated results. One of the limitations is the randomness of the results, such that we are not able to really determine the nature or the real performance of the simulations with different grid sizes due to it. One solution towards this problem can be to test with a larger number of runs that mitigate this problem to a certain level. Another limitation is that analysing based on the numbers of increasing grid size isn't sufficient in terms of the entire simulation, hence future work is expected to test on some other factors like the number of base station iterations, increasing the threshold value, etc…

MONASH University
Information Technology

References

A. Gamage, B. M. S. V., & Baskaran, V. M. (2020). Simulation and Analysis of Distributed Wireless Sensor Network using Message Passing Interface. https://www.researchgate.net/publication/342655333_Simulation_and_Analysis_of_Distributed_Wireless_Sensor_Network_using_Message_Passing_Interface

B. Niethammer, C., & Rabenseifner, R. (2019). An MPI interface for application and hardware aware cartesian topology optimization. *Proceedings of the 26th European MPI Users' Group Meeting*, *6*, 1-8. https://doi.org/10.1145/3343211.3343217