Design and Implementation of an Alert Detection in a Distributed Wireless Sensor Network

Bryan Ho Yung Kynn[1], Chew Shen Min[2]

School of Information Technology
Monash University
Malaysia

[1]bhoo0005@student.monash.edu
[2]sche0094@student.monash.edu

***Word Count*—3,499 words**

***Abstract*—Wireless Sensor Networks (WSN) have been widely used and applied in environmental monitoring which includes agricultural monitoring, habitat monitoring, greenhouse monitoring, climate monitoring and forest monitoring [1] to assist in reducing cost and time for resolving environmental issues. This report aims to combat rapid deforestation through open air burning by proposing and analysing an alert detection simulation using a 2D Cartesian grid topology of sensor nodes with a common base station. The system allows for a dynamic grid initialization in which the row, column and number of nodes are specified by the user. The sensor nodes can communicate with their direct neighbours to compare temperature readings and inform the base station for potential alerts given a certain threshold is exceeded. The base station will compare the received values with the information provided by the infrared satellite to determine if the alert is positive or negative, then records the results to a text file. To design the simulation, the Message Passing Interface (MPI) library in C [2] is used for communication and parallelizing the algorithm.**

***Keywords*—wireless sensor networks, Cartesian grid topology, inter process communication, message passing interface, IPC, MPI, WSN**

I. Introduction

In recent years, the issue of forest fires that transpired in distant parts of the globe such as Australia, Russia and the United States of America (USA) have resulted in increased global awareness and concern on the profound consequences of destructive conflagrations to the public health [3]. As a result, governmental and non-governmental organizations are collaborating to incorporate ground based wireless sensors to detect abnormalities in the surface temperature for actionable intelligence.

This work attempts to simulate a wireless sensor network (WSN) and satellite network for alert detection through an Inter Process Communication (IPC) architecture. An IPC is a mechanism that allows the exchange of data between processes [4]. The motive for implementing the simulation in a 2D Cartesian grid topology where sensor nodes communicate between their adjacent neighbours is because this architecture offers scalability and efficiency [5].

Assuming the clock is synchronised between both the sensor nodes and the base station, a hypothesis introduced to this implementation is that for an increasing number of simultaneous alert reports sent by the sensor nodes to the base station, due to the alerts being queued are only addressed by the base station on subsequent iterations, the communication time will increase. This would also potentially cause the base station to no longer match the temperature values received by the infrared imaging satellite.

The design and implementation are such that a dynamic 2D Cartesian grid-based architecture with a single base station is initialized. Sensor nodes will alert the base station given its own temperature exceeds a given threshold and comparisons made amongst its neighbours are within a certain tolerance range. The base station is designed to improve the accuracy of the reports where it has its own table of temperatures randomly generated using POSIX threads as a simulation of the heatmap information provided by the infrared imaging satellite, which is used for secondary comparisons with the received temperature readings to validate the claim. This is followed up with an analysis of the results to investigate the communication delays incurred by an increasing rate of simultaneous alert reports in a single iteration.

This implementation is possible using the Message Passing Interface (MPI) library. MPI provides functions to initialize a Cartesian communicator, perform dynamic Cartesian mapping and compute Cartesian shifting. This essentially enables the ability to group sensor nodes in a single communicator endowed with a 2D Cartesian grid topology that allows them to query the communicator about their neighbours.

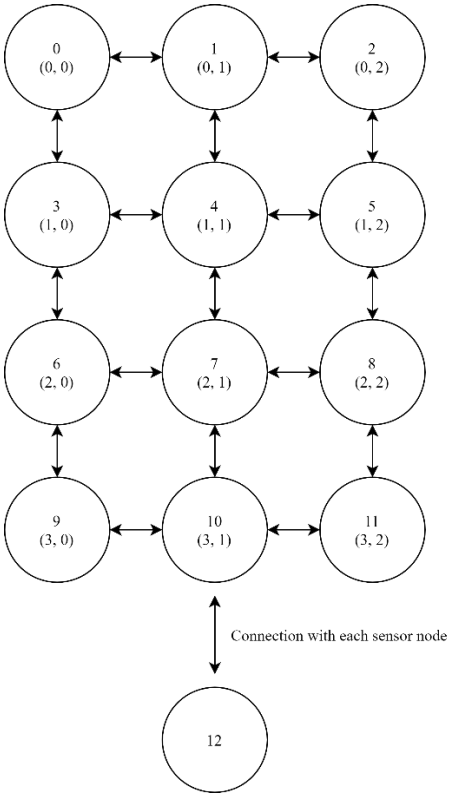
In the following sections, Section II describes the design scheme of the architecture with pseudocode implementations, Section III analyses the behaviour of the simulator and finally, Section IV concludes the report.

II. Design Scheme for Sensor Network

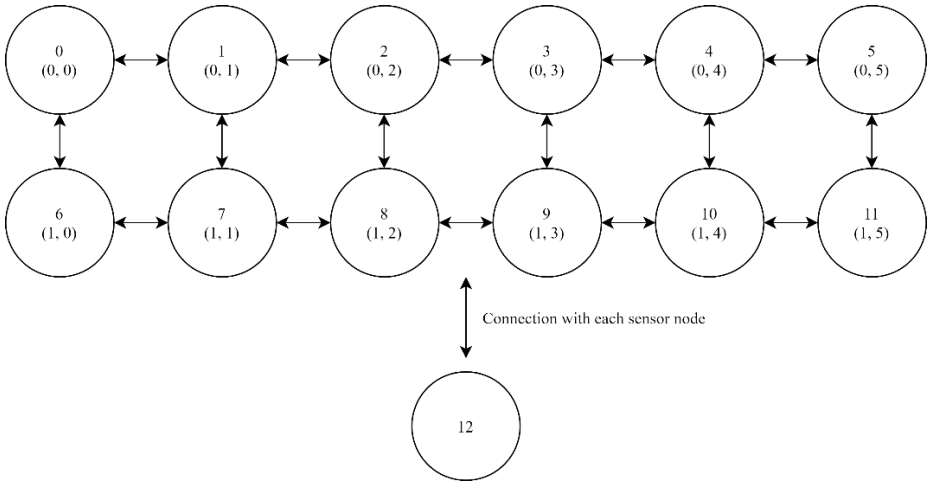
A. Inter Process Communication Grid Architecture

The WSN uses a 2D Cartesian grid-based 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.

Algorithm 1 Splitting the processes where the last rank is the base station while the rest are sensor nodes.

```
1:  procedure SPLITCOMM(comm)
2:      my_rank, size ← 0
3:      coord ← {0, 0}
4:      new_comm ← NULL
5:      MPI_ COMM_SIZE(comm, size)
6:      MPI_COMM_RANK(comm, my_rank)
7:      MPI_COMM_SPLIT(comm, my_rank == size - 1, 0,
8:  new_comm)
9:      if my_rank == size - 1 then
10:         BASESTATION(comm, new_comm)
11:     else
12:         SENSORNODES(comm, new_comm)
```

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.

B. Sensor Nodes

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

```
1:  procedure SENSORNODES(comm, new_comm)
2:      my_rank, size, exit , time_taken ← 0
3:      MPI_COMM_SIZE(comm, size)
4:      MPI_COMM_RANK(comm, my_rank)
5:      comm2D ← CREATE_CARTESIAN_TOPOLOGY()
6:     nbr_i_lo, nbr_i_hi, nbr_j_lo, nbr_j_hi ← 0
7:      MPI_CART_SHIFT(comm2D, 0, 1, nbr_i_lo,
8:  nbr_i_hi)
9:      MPI_CART_SHIFT(comm2D, 1, 1, nbr_j_lo,
10: nbr_j_hi)
11:     send_request, receive_request, send_status,
12: receive_status ← {0, 0, 0, 0}
13:     SRAND(TIME(NULL) + my_rank)
14:     while true do
15:         MPI_RECV(&exit, 1, MPI_INT,
16: MPI_ANY_SOURCE, MPI_ANY_TAG, comm,
17: &status)
18:         if my_rank != size -1 then
19:             MPI_SEND(&exit, 1, MPI_INT, my_rank + 1,
20: 0, comm)
21:         if exit then break
22:         rand_temperature ← (RAND() % (90 – 50 + 1)) + 50
23:         neighbours ← {nbr_i_lo, nbr_i_hi, nbr_j_lo,
24: nbr_j_hi}
25:         recv_temperature ← {-1, -1, -1, -1}
26:         for i ∈ [1, … , 4] do
27:             MPI_ISEND(rand_temperature, 1, MPI_INT,
28: neighbours[i], 0, new_comm, send_request[i])
29:             MPI_IRECV(recv_temperature[i], 1, MPI_INT,
30: neighbours[i], 0, new_comm, receive_request[i])
31:         MPI_WAITALL(4, send_request, send_status)
32:         MPI_WAITALL(4, receive_request,
33: receive_status)
34:         if rand_temperature >= 80 then
35:             count ← 0
36:             for i ∈ [1, … , 4] do
37:                 if ABS(rand_temperature –
38: recv_temperature[i]) <= 5 then count ← count + 1
39:             if count >= 2 then
40:                 report ← {my_rank, nbr_i_lo, nbr_i_hi,
41: nbr_j_lo, nbr_i_hi, rand_temperature,
42: recv_temperature[0], recv_temperature[1],
43: recv_temperature[2], recv_temperature[3], time_taken}
44:                 MPI_SEND(report, 11, MPI_INT, size - 1, 0,
45: comm)
```

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 temperature value at random.

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

A threshold of +5/-5 from the current node's temperature reading is set to verify that it is a result of open burning. 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 temperature readings, reporting node temperature reading and reporting node's rank. The array of information would then be reported to the base station using **MPI_Send**.

C. Infrared Imaging Satellite

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

```
1:  procedure *GENERATETEMPERATURE(*pArg)
2:      num_nodes ← *((int*) pArg)
3:      i ← NULL
4:      for i ∈ [0, … , num_nodes - 1] do
5:          heatmap[i] ← (RAND() % (90 – 50 + 1)) + 50
6:      return NULL
```

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

The infrared imaging satellite's purpose being to provide temperature readings of the surface where the sensor nodes are located (e.g. aerial based surface temperature readings). These temperature readings will then be used for secondary comparisons with the reports received from the sensor nodes within a given time window.

Temperature 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.

D. Base Station

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

```
1:  procedure BASESTATION(comm, new_comm, iter,
    nrows, ncols)
2:      *pWriteFile, *pKill ← NULL
3:      tid ← NULL
4:      i, size, num_nodes, alert, exit, kill,
5:  adjacent_matches, total_msg, total_true_alerts,
6:  total_false_alerts, total_msg_between, flag ← 0
5:      satelliteTime[1, …, 128] is empty array
6:      timeInDateTime[1, … , 128] is empty array
7:      alertTime[1, … , 128] is empty array
8:      count ← 1
9:      report[1, … , 11] is empty array
10:     commTime, time_taken ← 0
11:     start, end ← 0
12:     CLOCK_GETTIME(CLOCK_MONOTONIC, &start)
13:     status ← 0
14:     MPI_COMM_SIZE(comm, &size)
15:     num_nodes ← size - 1
16:     heatmap ← MALLOC((num_nodes) * SIZEOF(int))
17:     pWriteFile ← FOPEN("log.txt", "w")
18:     while iter > 0 do
19:         MPI_SEND(&exit, 1, MPI_INT, 0, 0, comm)
20:         PTHREAD_CREATE(&tid, 0, generate_temperature,
21: &num_nodes)
22:         PTHREAD_JOIN(tid, NULL)
23:         CONVERTTOTIMESTAMP(satelliteTime, 128)
24:         MPI_IPROBE(MPI_ANY_SOURCE,
25: MPI_ANY_TAG, comm, &flag, &status)
26:         total_msg_between ← 0
27:         adjacent_matches ← 0
28:         if flag then
29:             CONVERTTOTIMESTAMP(alertTime, 128)
30:             MPI_RECV(report, 11, MPI_DOUBLE,
31: MPI_ANY_SOURCE, MPI_ANY_TAG, comm,
32: &status)
33:             commTime ← MPI_Wtime() – report[10]
34:             total_msg_between ← total_msg_between + 1
35:             total_msg ← total_msg + 1
36:             CONVERTTOTIMESTAMP(timeInDateTime, 128)
37:             if (ABS(report[5] – heatmap[report[0]]) <= 5)
38: then
39:                 total_true_alerts ← total_true_alerts + 1
40:             else
41:                 total_false_alerts ← total_false_alerts + 1
42:             for i ∈ [0, … , NEIGHBOURS-1] do
43:                 if report[i + 1] != -2 then
44:                     adjacent_matches ← adjacent_matches + 1
45:             WRITETOLOGFILE(pWriteFile, report, count,
46: commTime, satelliteTime, total_msg_between,
47: adjacent_matches)
48:         pKill ← FOPEN("kill.txt", "r")
49:         FSCANF(pKill, %d, &kill)
50:         FCLOSE(pKill)
51:         if kill then break
52:         iter ← iter – 1
53:         count ← count + 1
54:     exit ← 1
55:     MPI_SEND(&exit, 1, MPI_INT, 0, 0, world_comm)
56:     clock_gettime(CLOCK_MONOTONIC, &end)
57:     time_taken ← (end.tv_sec – start.tv_sec) * 1e9
58:     time_taken ← (time_taken + (end.tv_nsec –
59: start.tv_nsec)) * 1e9
60:     WRITESUMMARYTOLOGFILE(time_taken,
61: total_msg, total_true_alerts, total_false_alerts)
62:     FREE(heatmap)
63:     FCLOSE(pWriteFile)
64:     return 0
```

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 infrared temperature values from the infrared imaging satellite periodically. **pthread_create** creates a POSIX thread which would run the function defined by Algorithm 3 to generate a random temperature 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, **flag** would be set to 1 by default which would trigger our while loop to process the alert or message.

3) Alert Processing – Once **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 temperature reading would be compared to the infrared temperature reading from the infrared imaging satellite to identify for 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 infrared reading will not match the initial reading.

III. RESULTS AND DISCUSSION

The runs were executed and completed on personal computers each with 1000 iterations.

A. Summary of implementation to obtain results

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

2) Base station compares temperature values of each alert with the infrared temperature 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.

Figure 2 displays the information that the base station would be responsible to log into the text file.

```
-----------------------------------------------------
Iteration: 1000
Logged Time:                Wed 2020-10-28 18:28:32
Alert Reported Time:        Wed 2020-10-28 18:28:32
Alert Type: False

Reporting Node   Coordinates     Temperature
11               (1, 5)          81

Adjacent Nodes   Coordinates     Temperature
5                (0, 5)          81
10               (1, 4)          83

Infrared Satellite Reporting Time: Wed 2020-10-28 18:28:32
Infrared Satellite Reporting (Celsius): 74
Infrared Satellite Reporting Coordinates: (1,5)

Communication Time (seconds): 0.935178
Total Messages send between reporting node and base station: 1
Number of adjacent matches to reporting node: 2
```

Fig 2: Base Station log file for event details

1) Base Station Log File – The base station would log two types of information based on the information sent from the sensor nodes which are the details of the alert event and the details of the logged event.

Figure 2 is an example of an iteration of an alert event logged by the base station. This event is triggered at the 1000th iteration of the simulation as shown in the first line, the iteration number. The following two timestamps which records the logged time of the alert and the time of alert from the sensor node upon receiving the alert at the base station. In this example, the simulation of the base station and the sensor nodes are all in the same machine which is why we are not seeing a difference in times. However, this feature is useful when the occasion arises where the sensor nodes and base station are located at two different geographical locations where message travel time is more significant.

The next five lines would state the details of the sensor node and its adjacent neighbour sensor nodes that were compared. Reporting Node would be the sensor node that reported the alert along with the sensor node's coordinates in the Cartesian topology stated in Coordinates and the temperature reading of the node stated under Temperature. Adjacent Nodes are the neighbour nodes that were compared with the Reporting Node to identify if the temperature reading is an alert. The adjacent nodes must meet a threshold where the difference between the reported node temperature must not be less than -5 or more than +5 of the reported node temperatures.

The next three lines states the recorded information from the infrared imaging satellite of the reported node. It includes the Reporting Time of the infrared imaging, Reporting Temperature in Celsius and the coordinates of the infrared imaging of the reporting node. Lastly, communication time and total messages sent between the sensor node and the base station are also recorded along with the adjacent matches of the reporting node.

```
Summary

Total number of messages received: 999
Total number of true alerts: 469
Total number of false alerts: 530
Total communication time: 6.831649
```

Fig 3: Summary of all events recorded by 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.

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 infrared imaging satellite.

B) Communication Time – The communication time includes event detection from the sensor nodes after every iteration, receiving the message and temperature comparisons between the temperatures of the reporting node and the infrared imaging 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.

IV. Conclusion

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 infrared temperature 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.

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