MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
29691931 (50%)	Spasic	Pavle
29605970 (50%)	Zhang	Yixuan

* Please include the names of all other group members.

Unit name and code	Parallel Computing – FIT3143	
Title of assignment	Assignment 2: Report title: Simulating Wireless Sensor Networks within the MPI Library for Alert Detection	
Lecturer/tutor	ABM Russel/ Zhinoos Razavi/ Haomai Li	
Tutorial day and time	Friday 4pm	**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 31/10/2020		**Date submitted** 31/10/2020

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.

Signature Pavle Spasic Date: 31/10/2020 Signature Yixuan Zhang Date: 31/10/2020

Signature ____________________Date:____________________ Signature ________________________________Date:________

Signature ____________________Date:______________________ Signature ________________________________Date:________

Simulating Wireless Sensor Networks within the MPI Library for Alert Detection

Pavle Spasic
Faculty of Information
Monash University, Clayton
Melbourne, Australia

Yixuan Zhang
Faculty of Information
Monash University, Clayton
China

Words - 2113

***Abstract*— The need to oversee and manage open air burning has come to an implementation of a wireless network system. Including ground sensors in a cartesian grid system along with a base station and an infrared satellite. In the grid, the ground sensors are allowed to communicate with their adjacent neighbors and, when appropriate, report to the base station which compares data with the infrared satellite and log a report. In this report we explore the simulation of such a system through the MPI library and POSIX threads through synchronous and asynchronous communication. This simulation allows us to investigate effects of the structure of the cartesian topology on the performance of the distributed system. Finding that dimensions of the cartesian grid did not have a major impact on the communication times between nodes within the system, and rather the data showing a potential factor being the number of reports affecting the communication times.**

Keywords—MPI, simulate, wireless sensor networks, alert detection, cartesian topology, POSIX thread

I. Introduction (*Heading 1*)

The ability to monitor open air burning is becoming increasingly important to manage the deforestation that is caused. A solution has been devised to establish a wireless sensor network connecting ground sensors to satellites and a base station. When ground sensors record a temperature above a threshold, it compares its value with other adjacent sensors within a cartesian grid, if 2 or more adjacent sensors are within a defined range a report is sent to the base station. The base station then compares the received report with information from an infrared imaging satellite before contacting authorities to control a potential fire.

Studying and monitoring the performance of a wireless network system is important during the development of the system, although in doing so may require more effort and financial requirements than needed (Khemapech, Miller & Duncan, 2005). There are several methods to simulate wireless network systems (Khemapech et al., 2005), although in this report we simulate the wireless network through the use of the MPI library.

Through the MPI library we can have separate MPI processes simulate the base station and ground sensors. The MPI library also allows the use of a cartesian topology to simulate the grid formation of the ground sensors. The information from the infrared satellite is simulated through the use of a POSIX thread.

Our goal with the report is to explore the effect of the dimensions of the grid on the communication times. The dimensions can be defined as m x n = s, where m and n are the width and height respectively, and s being the total number of systems or processes in this simulation. We hypothesize that grid sizes that have a square root will be most efficient at which the dimensions are equal, such that m = n. We will analyse the communication times of each to draw a conclusion.

The report aims to explore the simulation of the wireless network system within the MPI library and how the dimensions affect the performance. In the following sections we describe the design and implementation of the MPI simulation, analyse the performance of the architecture and the effect of the different dimensions, then come to a conclusion. that follow.

II. Design scheme and implementation in MPI

The algorithm of assignment mainly consists of three parts:

Part 1. Split the all processes into two parts, one part contains only one node which is the base station, the rest of the nodes are made into a cartesian grid.

Part 2. When Sensor nodes run , the time period is 2 seconds per iteration, sensor nodes check their own temperatures, if it is greater than the threshold, it will request messages from its neighbour nodes to compare their temperatures with their own temperature, In addition, the comparison got a tolerance value, if the difference between temperatures exceed the tolerance value, set a same_count to count the times that temperatures exceeding the tolerance. If same_count>=2, send a warning alert to the base station.

Part 3. At the same time, base station also receive the messages send from nodes (again 2 seconds per iteration), store the data to a n*m array called infos. Then for each cell in recv[ ], compare the received alert data with temperatures obtained from the infrared satellite, then, write contents to the log file according to its corresponding information.

We will use 3 subsections to demonstrate the algorithm.

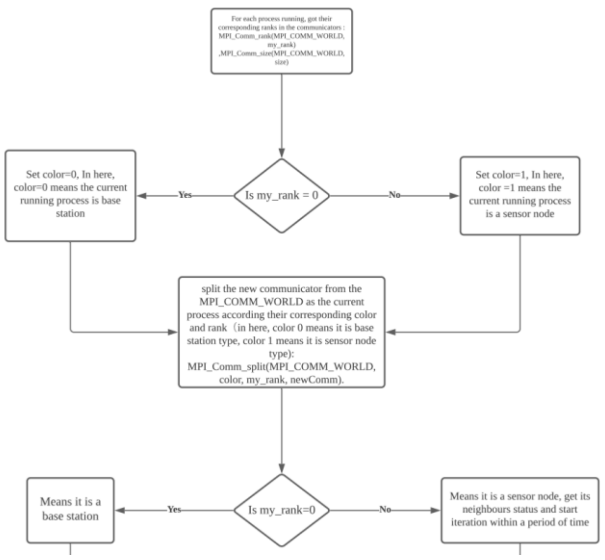

For each process running, got their corresponding ranks in the communicators : MPI_Comm_rank(MPI_COMM_WORLD, my_rank) ,MPI_Comm_size(MPI_COMM_WORLD, size)
Set color=0, In here, color=0 means the current running process is base station
Yes
Is my_rank = 0
No
Set color=1, In here, color =1 means the current running process is a sensor node
split the new communicator from the MPI_COMM_WORLD as the current process according their corresponding color and rank（in here, color 0 means it is base station type, color 1 means it is sensor node type): MPI_Comm_split(MPI_COMM_WORLD, color, my_rank, newComm).
Means it is a base station
Yes
Is my_rank=0
No
Means it is a sensor node, get its neighbours status and start iteration within a period of time

Figure 1 Subsection of Flow Chart

A. *Subsection 1*

This block of flowchart mainly introduces the Part 1 above, which is to split the nodes into two parts, one part contains only one node which is the base station, the rest of the nodes are made into a cartesian grid. In here, nodes with rank 0 are set to be the base station, the nodes with other rank are sensor nodes. When split the nodes, firstly check whether my_rank = 0 (to identify whether the running process is a base station or not), After nodes are distinguished, we set color 0 to base station type, set color 1 to sensor nodes type, then MPI_Comm_split can be used. Hence all nodes are split into two types.

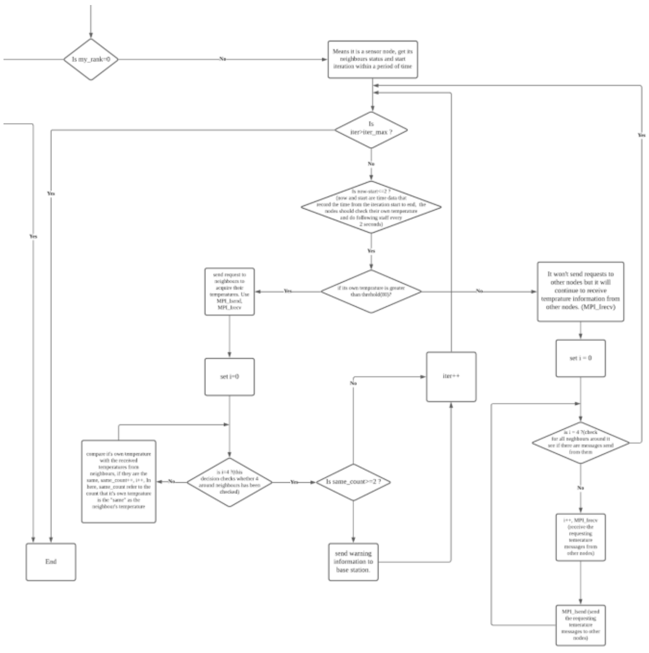

Is my_rank=0
No
Means it is a sensor node, get its neighbours status and start iteration within a period of time
Is iter>iter_max ?
Yes
Yes
Yes
No
Is now-start<=2 ? (now and start are time data that record the time from the iteration start to end, the nodes should check their own temperature and do following staff every 2 seconds)
Yes
send request to neighbours to acquire their temperatures. Use MPI_Isend, MPI_Irecv
Yes
if its own temprature is greater than threhold(80)?
No
It won't send requests to other nodes but it will continue to receive temprature information from other nodes. (MPI_Irecv)
set i=0
iter++
set i = 0
No
is i = 4 ?(check for all neighbours around it see if there are messages send from them
compare it's own temperature with the received temperatures from neighbours, if they are the same, same_count++, i++, In here, same_count refer to the count that it's own temprature is the "same" as the neighbour's temperature
No
is i=4 ?(this decision checks whether 4 around neighbours has been checked)
Yes
Is same_count>=2 ?
No
End
send warning information to base station.
i++, MPI_Irecv (receive the requesting temerature messages from other nodes)
MPI_Isend (send the requesting temerature messages to other nodes)

Figure 2 Subsection 2 of Flow Chart

B. *Subsection 2*

This block of flowchart introduced the Part 2 described above which is about sensor nodes, firstly, for each sensor node, integer value iter is set to be 0 anditer_max is been set, that means all sensor nodes can't have a number of iterations exceeding this value. At start, check whether current node's iteration is greater than iter_max, if it is, process will end immediately (MPI_Finalize), if not, it will set a particular period of time for each iteration, in here, we set 2 seconds as fixing time period for each node (2 seconds per iteration). Then we will check whether the current node's temperature is greater than threshold(80):

If the current node's temperature is greater than 80, then this node will send a request to its neighbours and receive neighbours' temperatures for comparison. If there are more than 2 nodes sent to temperatures that are the "same" (with 3 tolerance) as its own, it will send an alert to the base station. In here we use a for loop to make sure all neighbours' data has been compared and use same_count to record the number of "same" temperatures.

If the current node's temperature is less than 80, it will not send requests to neighbours, but it will need to wait for requests from other nodes (other nodes may need it to send its temperature for comparison). Use MPI_Irecv and MPI_Isend to implement it.

After all above has been completed, it will increase the iter by 1 and start a new iter, repeat all steps above until iter>iter_max, then it will end.

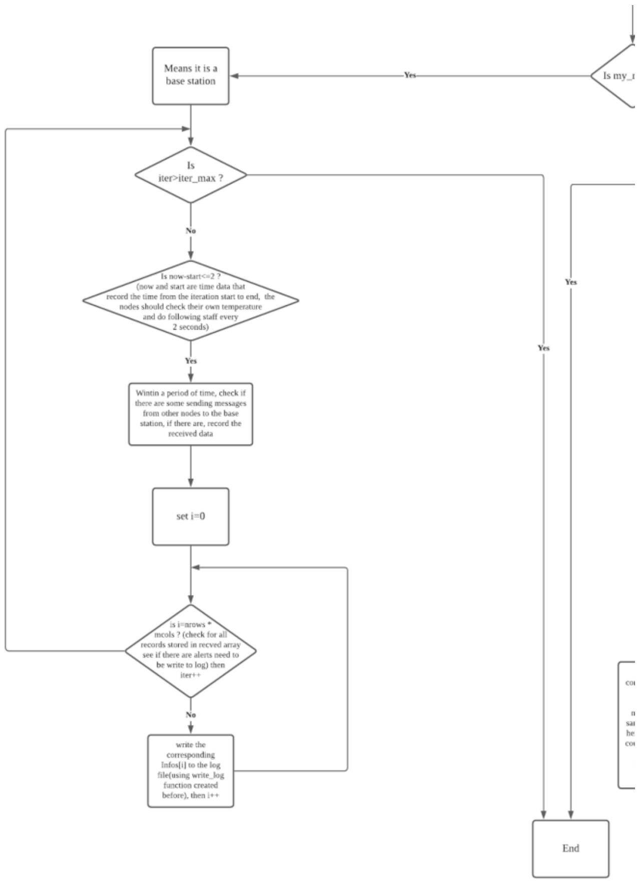

Means it is a base station
Yes
Is my_r
Is iter>iter_max ?
No
Is now-start<=2 ? (now and start are time data that record the time from the iteration start to end, the nodes should check their own temperature and do following staff every 2 seconds)
Yes
Yes
Yes
Wintin a period of time, check if there are some sending messages from other nodes to the base station, if there are, record the received data
set i=0
is i=nrows * ncols ? (check for all records stored in recved array see if there are alerts need to be write to log) then iter++
No
write the corresponding Infos[i] to the log file(using write_log function created before), then i++
End

Figure 3 Subsection 3 of Flow Chart

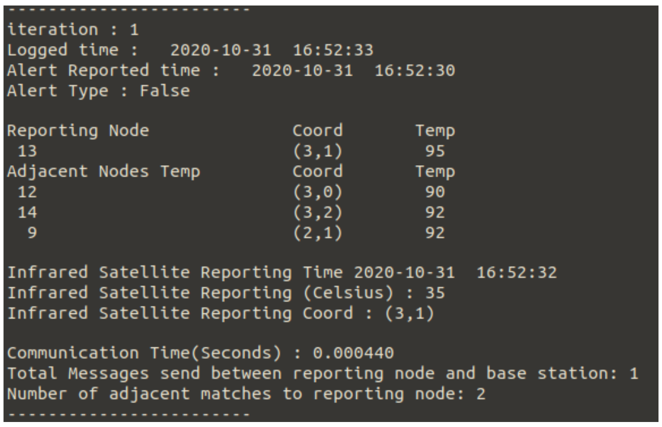
C. *Subsection 3*

This block of flowchart is about Part 3 described above, For base station, similar to sensor nodes, it also got iter, iter_max, and a particular period of time, the function of them are similar as above. Info* infos = (Info*)malloc(sizeof(Info) * ncols * nrows), infos is created to record the data got from sensor nodes. Base station will wait for every sensor node to send their alert, after receiving all data from nodes, it will compare the temperature data stored in infos[ ] with the data from infrared satellites. In addition, logged_time, infared_time, alert_time are all recorded and written into the log file. After all this, the base station will start a new iteration and repeat the steps above until iter>iter_max.

III. Results and Discussion

Simulation and testing was done on a virtual machine run on an Intel® Core(™) i7-4770 CPU @ 3.40 GHz with 4 cores and 8 logical processes. In order to test cartesian grid sizes greater than 4, the oversubscribe command was used to allow more than 4 MPI processes. To gather information, 5 test runs of 17 processes with varying grid dimensions were run, 1x16, 2x8, 4x4, 8x2 and 16x1. Each test run went through 20 iterations to ensure enough data, as grid dimensions such as 16x1 report less more square grids. The simulation was set with a threshold of 80 degrees and a neighbour range of 3.

The performance of the cartesian grid dimension was assessed using the communication time between the ground sensors and the base station.



```
------------------------
iteration : 1
Logged time :   2020-10-31  16:52:33
Alert Reported time :   2020-10-31  16:52:30
Alert Type : False

Reporting Node              Coord       Temp
 13                         (3,1)        95
Adjacent Nodes Temp         Coord       Temp
 12                         (3,0)        90
 14                         (3,2)        92
  9                         (2,1)        92

Infrared Satellite Reporting Time 2020-10-31  16:52:32
Infrared Satellite Reporting (Celsius) : 35
Infrared Satellite Reporting Coord : (3,1)

Communication Time(Seconds) : 0.000440
Total Messages send between reporting node and base station: 1
Number of adjacent matches to reporting node: 2
------------------------
```

Figure 4 Base station log file

A. *Base Station log file*

When each node encounters an alert it is sent to the base station which logs its information and its comparison to the Infrared Satellite. This section will go over the contents in the log file.

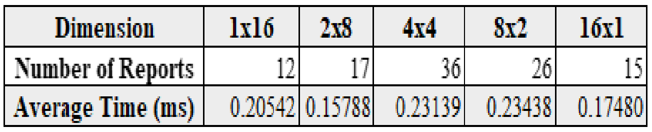
Figure 4 shows which iteration the alert was called in the first line. The next two lines show the time and date when the event was logged to the file, and when the base station was reported to by the ground sensor respectively. As the simulation has the ground sensor and the base station on the same machine, the difference in the times are small and this information is important, although, in the wireless application, the geographical distance may affect information transmission rate. The line denotes the alert type of the report, True is if the temperature from the ground sensor matches the temperature from the Infrared Satellite, and False otherwise.

This next part of the log file shows which node in the cartesian grid called the alert to the base station, its coordinates and its temperature. The next few lines show the same information on the adjacent nodes in the cartesian grid. The number of lines can vary depending on the location of the node in the grid. The 3 after this shows the information the base station received from the infrared satellite, the time the information was received, the temperature recorded and the coordinates to match the reported node respectively.

The final 3 lines give information on the report sent from the ground sensors, the time between the ground sensor sending the information and when the base station received it, the number of messages received by the base station from the ground sensor, and the amount of adjacent nodes that fell within the range of the initial sensor respectively. The communication time here is the basis of the performance analysis of the different cartesian grid dimensions.

B. *Equations*

Figure 5 shows the resulting average communication times of different dimensions of a 16 processor grid in milliseconds.



Dimension	1x16	2x8	4x4	8x2	16x1
Number of Reports	12	17	36	26	15
Average Time (ms)	0.20542	0.15788	0.23139	0.23438	0.17480

Figure 5 Test result of varying dimensions

Earlier in this report we hypothesized that for cartesian grid where n x m = s, with n and m being width and height respectively and s being the total processors in the grid, that the most efficient dimension, lowest average communication time, would be one with n = m. However, through testing, this is not the case, the 4x4 layout had the 2 largest average time of the 5 dimensions tested. Although 20 iterations were complete, due to the randomness allot can vary with the number of reports.

That also leads into another observation where rather the performance being dependent on the dimension of the grid, and more on the total number of reports instead. 2x8 has a similar dimension to 8x2, and the simulation happened to have more reports on the 8x2 and the average time shows a similar relationship. Although, in the case of 1x16 and 16x1, the same cannot be said. In this simulation the communication times are very small, and in addition to the small amount of iterations, causes very close average times between all the dimensions. In a real world wireless network setup, the communication times may be larger so a pattern may be easier to find between the dimensions.

IV. Conclusion

Although, as mentioned earlier, a conclusion was drawn that the dimensions of the cartesian grid doesn't have a noticeable effect on the performance of communication times. Although, due to the randomness of this simulation, a larger sample size may show more telling results.

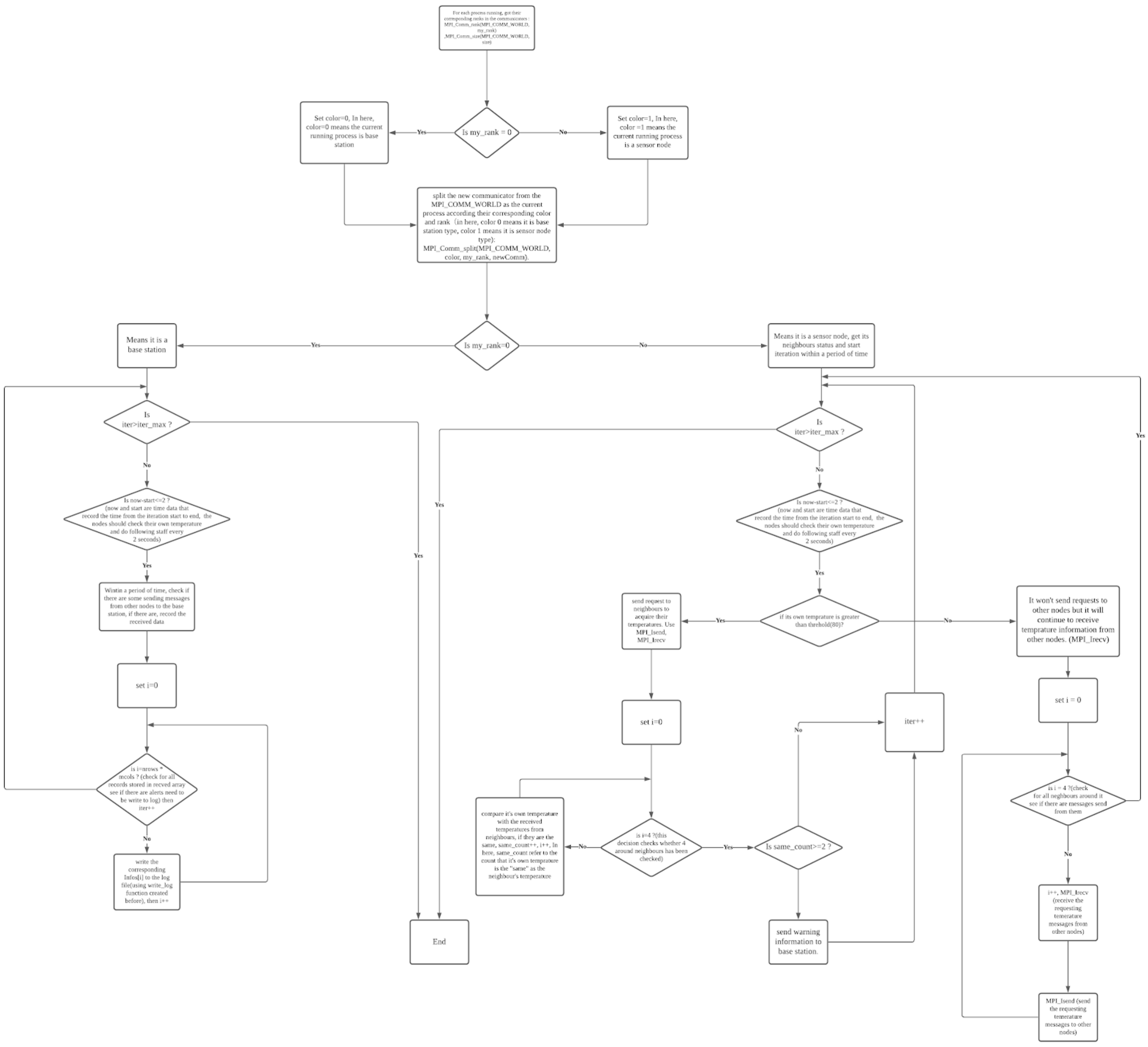
For future renditions of this report, the implementation of an automated entry of communication times will allow for much larger sample sizes and iterations, or potentially having a fixed set of numbers to remove any random aspect, to truly just analyse the effect that dimensions have on the system. Also, the data we did have showed a potential avenue where the communication times can rather be based on the number of reports over time or iterations, this may prove helpful for a future investigation into finding an optimum structure.

References

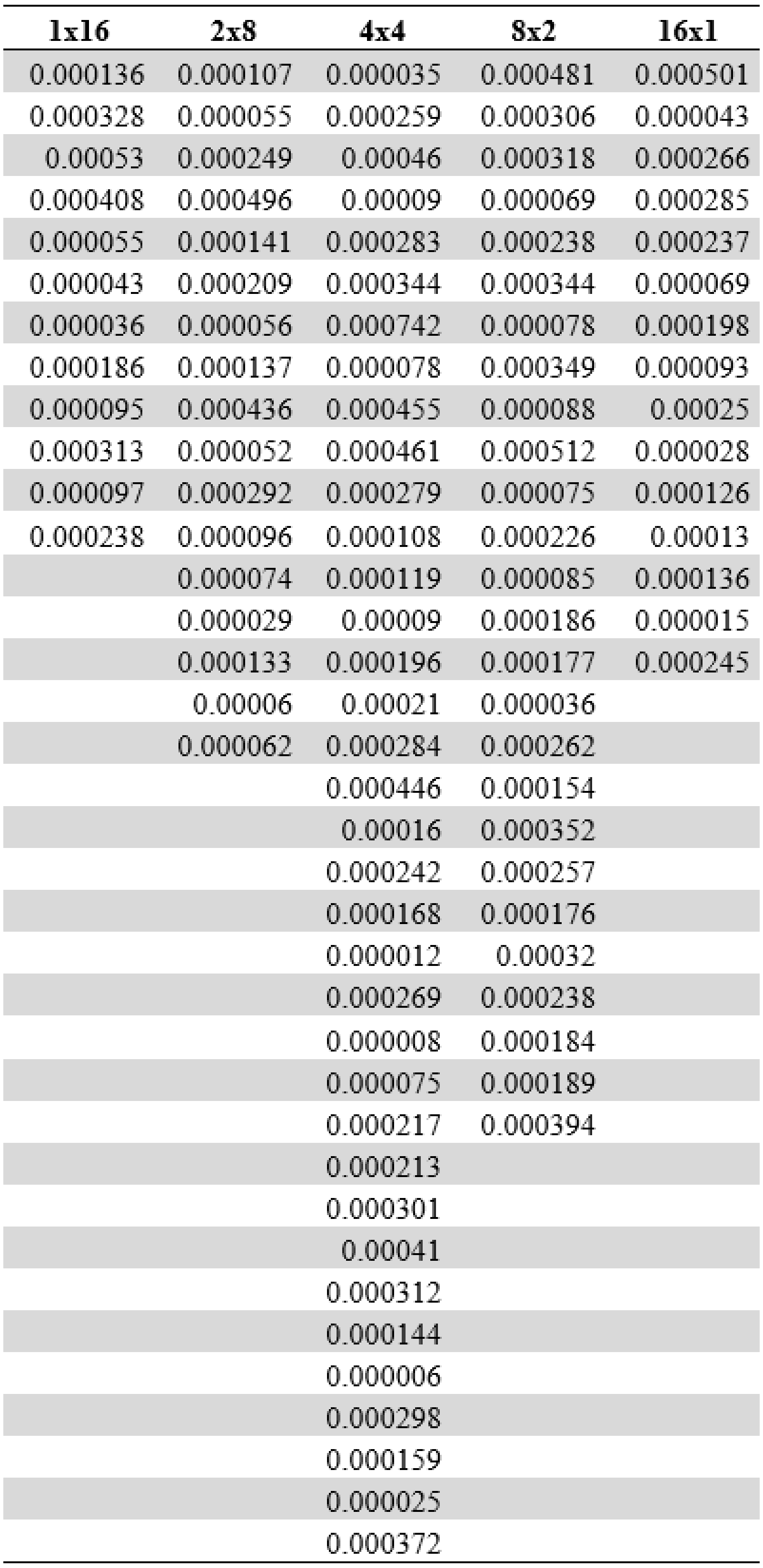
[1] Khemapech, I., Miller, A., & Duncan, I. (2005). Simulating Wireless Sensor Networks. Rertieved from https://www.researchgate.net/profile/Alan_Miller5/publication/228614879_Simulating_wireless_sensor_networks/links/00463525ee353dff66000000.pdf

APPENDIX

Full flow chart


For each process running, got their corresponding ranks in the communicators : MPI_Comm_rank(MPI_COMM_WORLD, my_rank) ,MPI_Comm_size(MPI_COMM_WORLD, size)
Is my_rank = 0
Yes
No
Set color=0, In here, color=0 means the current running process is base station
Set color=1, In here, color =1 means the current running process is a sensor node
split the new communicator from the MPI_COMM_WORLD as the current process according their corresponding color and rank（in here, color 0 means it is base station type, color 1 means it is sensor node type): MPI_Comm_split(MPI_COMM_WORLD, color, my_rank, newComm).
Is my_rank=0
Yes
No
Means it is a base station
Is iter>iter_max ?
No
Is now-start<=2 ? (now and start are time data that record the time from the iteration start to end, the nodes should check their own temperature and do following staff every 2 seconds)
Yes
Wintin a period of time, check if there are some sending messages from other nodes to the base station, if there are, record the received data
set i=0
is i=nrows * mcols ? (check for all records stored in recved array see if there are alerts need to be write to log) then iter++
No
write the corresponding Infos[i] to the log file(using write_log function created before), then i++
Yes
Yes
End
Means it is a sensor node, get its neighbours status and start iteration within a period of time
Is iter>iter_max ?
No
Is now-start<=2 ? (now and start are time data that record the time from the iteration start to end, the nodes should check their own temperature and do following staff every 2 seconds)
Yes
if its own temprature is greater than threhold(80)?
Yes
No
send request to neighbours to acquire their temperatures. Use MPI_Isend, MPI_Irecv
set i=0
compare it's own temperature with the received temperatures from neighbours, if they are the same, same_count++, i++, In here, same_count refer to the count that it's own temprature is the "same" as the neighbour's temperature
is i=4 ?(this decision checks whether 4 around neighbours has been checked)
No
Yes
Is same_count>=2 ?
No
send warning information to base station.
iter++
It won't send requests to other nodes but it will continue to receive temprature information from other nodes. (MPI_Irecv)
set i = 0
is i = 4 ?(check for all neghbours around it see if there are messages send from them
Yes
No
i++, MPI_Irecv (receive the requesting temerature messages from other nodes)
MPI_Isend (send the requesting temerature messages to other nodes)

Raw data from tests, all timings are in seconds



1x16	2x8	4x4	8x2	16x1
0.000136	0.000107	0.000035	0.000481	0.000501
0.000328	0.000055	0.000259	0.000306	0.000043
0.00053	0.000249	0.00046	0.000318	0.000266
0.000408	0.000496	0.00009	0.000069	0.000285
0.000055	0.000141	0.000283	0.000238	0.000237
0.000043	0.000209	0.000344	0.000344	0.000069
0.000036	0.000056	0.000742	0.000078	0.000198
0.000186	0.000137	0.000078	0.000349	0.000093
0.000095	0.000436	0.000455	0.000088	0.00025
0.000313	0.000052	0.000461	0.000512	0.000028
0.000097	0.000292	0.000279	0.000075	0.000126
0.000238	0.000096	0.000108	0.000226	0.00013
	0.000074	0.000119	0.000085	0.000136
	0.000029	0.00009	0.000186	0.000015
	0.000133	0.000196	0.000177	0.000245
	0.00006	0.00021	0.000036	
	0.000062	0.000284	0.000262	
		0.000446	0.000154	
		0.00016	0.000352	
		0.000242	0.000257	
		0.000168	0.000176	
		0.000012	0.00032	
		0.000269	0.000238	
		0.000008	0.000184	
		0.000075	0.000189	
		0.000217	0.000394	
		0.000213		
		0.000301		
		0.00041		
		0.000312		
		0.000144		
		0.000006		
		0.000298		
		0.000159		
		0.000025		
		0.000372		