Simulation and Analysis of Distributed Wireless Sensor Network using Message Passing Interface

School of Information Technology
Monash University
Malaysia

***Abstract*—Wireless Sensor Networks (WSN) are used by many industries from environment monitoring systems [1] to NASA's space exploration programs [2], as it has allowed society to monitor and prevent problems before they occur with less cost and maintenance. This document aims to propose and analyze an efficient inter process communication (IPC) architecture using a nearest neighbor/grid based socket architecture. A parallelized version of the AES encryption algorithm is also used in order to increase the security of the WSN. First the proposed architecture is compared and contrasted against other well established architectures. Next, the benefits and drawbacks of the AES encryption algorithm is elucidated. The Message Parsing Interface (MPI) library in C [3] is used for the communication while OpenMP [4] is used for parallelizing the encryption algorithm. Next an analysis is performed on the results obtained from multiple simulations. Finally a conclusion is made that the grid based IPC architecture with AES parallel encryption helps WSNs maintain security in communication while being cost and power efficient to operate.**

***Index Terms*—wireless sensor networks, wsn, interprocess communication, ipc, grid based architecture, socket architecture, nearest neighbor architecture, advanced encryption standard, parallel aes, openmp, mpi, distributed computing**

I. INTRODUCTION

The rapid growth of Internet of Things (IoT) devices and sensors has led to a high demand in efficient sensor architectures [1]. Since sensors mainly run on embedded devices and battery powered systems, the need for power efficient architectures are on the rise [5]. Currently, Wireless Sensor Network (WSN) architectures are the standard for low power sensor networks. To accomplish low power usage, each node in the WSN tries to reduce the amount of processing and communication done in them. In order to implement these WSNs, highly efficient Inter Process Communication (IPC) architectures are needed.

An Inter Process Communication is a system that allows data to be communicated between two processes [6]. Pipes, Socket, File, Signal, Shared Memory and Message Queue are few of the different IPC architectures currently available [7]. Since, any given node has to communicate with either another node or the base station and the nodes can be located in different geographical locations, out of the given architectures, the socket architecture will be the most suitable, as it creates end points on each node to send and receive data. There are different design space architectures for socket based IPC architectures such as hyper cube, pyramid, tree and grid (nearest neighbor). But since a sensor node can only communicate with the four adjacent nodes i.e. nearest neighbors, a grid based socket IPC architecture would be the best method for the WSN.

One might argue that the pyramid based architecture will also be efficient as it converges the number of nodes as it comes to the top of the pyramid and each level of the pyramid can communicate with their adjacent nodes. But the main issue with this approach is that it is complex to build and maintain, and the number of messages needed to send a message from a bottom level to the top level is high. This also means that there is more chance of failure and if one middle layer node fails, then the whole sub pyramid of that node will also fail. Therefore, using a grid based nearest neighbor architecture allows us to ensure that no catastrophic failures in the architecture can happen as failure in one sensor node does not effect other nodes around it. The vulnerability of the system failing when the base station fails, can be overcome by having a backup server or load balancer at the base station. Miller et al. [8] provides a more in-depth analysis on the benefits and drawback of each architecture.

Another major constraint of WSNs, is that not only do they have to operate on low power, but also be secure enough to broadcast sensitive sensor data. This is a major concern as most of the communication is done wirelessly and attackers can get the data easily [9]. To achieve this, state of the art efficient encryption standards are used by the system.

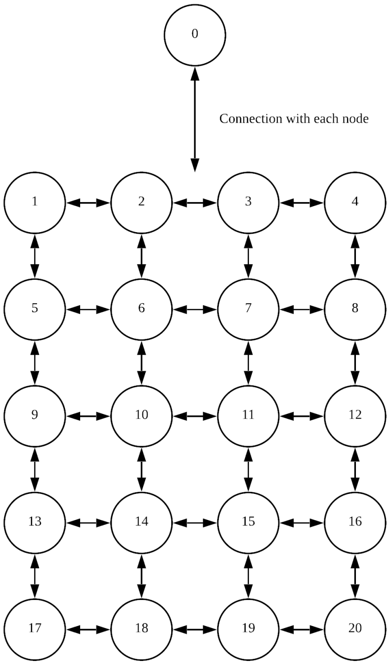
In the following sections, we explore and analyze *"How a grid based IPC architecture, that uses state of the art encryption standards can be made to run efficiently on a Wireless Sensor Network"*.

II. THEORETICAL ANALYSIS AND INTER PROCESS COMMUNICATION DESIGN

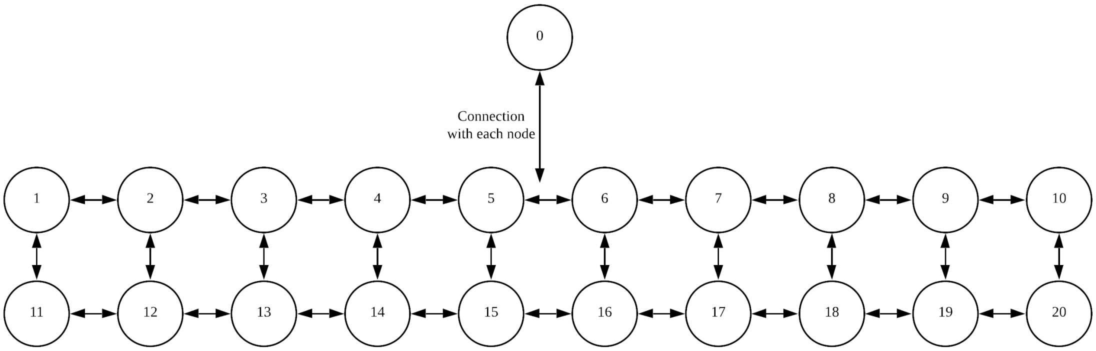
The IPC Design used by the WSN is discussed in this section. The WSN uses a grid based architecture where each node is able to communicate with the adjacent node and the base station only.

Fig. 1: Dynamic Grid Layout

(a) 4 X 5 grid



(b) 10 X 2 grid



A. *Inter Process Communication Grid Architecture*

The number of nodes used by the WSN depends on how many processes are allocated by the user at the start. If the user specifies the number of nodes as 21, then the system uses one node as the base and the rest as the sensor nodes. Using MPI, we allocate rank 0^{th} process to the base station and the rest of the nodes as the sensor nodes. The reason for selecting the base station as rank 0 is that only rank 0 get inputs from the user.

The system uses a dynamic grid based architecture. So the layout of the nodes are based on the height and width specified by the user. Figure 1 illustrates a simple example of how the user can rearrange a 21 node architecture. Figure 1a illustrates how the grid is initialized when the width and height is 4 and 5 respectively, while Figure 1b illustrates when its 10 and 2 respectively.

After initializing, each node is able to communicate with the adjacent four nodes i.e left, right, top and bottom nodes and also with the base station. The base station is able to communicate with each node individually. The adjacent nodes of each of the sensor nodes are calculated using the width and height specified by the user. Equation 1 and 2 can be used to determine the row index and column index of a given node in the WSN. *Note: Due to the base station being rank 0, we first minus 1 from the current rank*

$$rowindex = (rank - 1) \, // \, width \quad (1)$$

$$columnindex = (rank - 1) \ mod \ width \quad (2)$$

After determining the row index and the column index, we can use Equations 3, 4, 5 and 6 to get the four adjacent nodes.

$$left = rowindex \times width + columnindex \quad (3)$$

$$right = rowindex \times width + columnindex + 2 \quad (4)$$

$$top = (rowindex - 1) \times width + columnindex + 1 \quad (5)$$

$$bottom = (rowindex + 1) \times width + columnindex + 1 \quad (6)$$

Note : The shift in the ranks due to the base station in rank 0 is accounted for in the equations and also a check needs to be done to ensure that the current rank is not an edge or corner node.

Using this process, we do not need to send the rank values to the base station when an event occurs. The base station can use the rank of the incoming node to generate the four adjacent nodes in order. Therefore, this reduces the message size and optimizes the nodes to use less energy and the base to consume more energy for computation.

Hoefler et al. [10] mentions that the nearest neighbor architecture as one of the scalable process topologies. Therefore, this nearest neighbor algorithm used for WSN should be scalable as the number of sensors can increase with the growth in the field of IoT. Schneider and Hoefler then proposed methods to optimize Neighborhood Communications based on the 2D grid architecture we discussed above [11]. Miller et al. [8] also discussed about communication algorithms to be used with grid based architecture in broadcast and scatter/gather communication. Therefore, considering the prior research and the advantages it allows for WSNs, the 2D grid based nearest neighbor architecture will be used for our system.

B. Wireless Sensor Network Event Detection Algorithm

Now we look into the algorithm used by the system to detect events in the WSN. The system is divided into two main sections.

- Base Station which logs all the events
- Sensor Nodes which triggers and checks for events

Initially, the system generates the sensor network grid dynamically using the *width* and *height* given by the user which was discussed in Section II-A. Then both the sensor nodes and the base station start running their respective methods. First the IP Address and the MAC Address is sent from each node to be stored in the base station for logging purposes. These messages are encrypted using the AES encryption algorithm [12] discussed in section II-C. The base station will then decrypt the message and store it in a dynamic array. Below is the data passed from the node to the base station when initializing the system.

- IP Address of the node
- MAC Address of the node

Figure 2 breaks down the algorithm of the whole system in to a technical flowchart. Then Figure 3a explains how the system initializes the WSN, based on the dynamic user inputs. Now let us dive deep into the algorithms used by the base station and the sensor nodes to generate and detect events.

1) Base Station: After the WSN is initialized, the base station starts checking for messages from any node. If the user has entered -1 as input for the iteration, the system will terminate checking for events only if the "stop" keyword is entered by the user. In order for this to work, the base station creates a POSIX thread [13], which enables it to check for user inputs as well as log the events from the nodes. When the user inputs the "stop" keyword, the thread then sends a message to the nodes to stop event detection gracefully. Then the nodes will start sending the termination messages back to the base station and the system will shut down.

2) Sensor Nodes: Each node generates a random number and sends it to the adjacent nodes. This message which contains the random number, sent between the nodes are also encrypted. Then the sensor node starts receiving random numbers from the adjacent nodes. If three or more random numbers are the same, an event is detected. The algorithm also takes into account the random numbers generated by the nodes in the previous iteration. So it works as a sliding window with a time frame depth of 2. Then the nodes will notify the base station by sending a message. This message contains the following details:

- The random number that triggered the event
- An array of four elements which corresponds to the iteration of the matched value. If the left adjacent node matched with the random number of the previous iteration, the array will contain the iteration number of the previous element in the 0^{th} index of the array.
- The MPI Wtime of the event detection to be used to calculate the communication time
- The date time string of the time, when the event was detected to be used for logging

After sending the message to the server, it saves the random numbers from the current iteration. If the maximum number of iterations is reached or the user's stop signal is received, the node will send a termination signal to the base station to exit out of the system gracefully.

Non blocking send is used for the communication between the adjacent nodes, as blocking send has a chance of creating a deadlock. The communication between the sensor nodes and the base station is done using MPI blocking send and receive. This allows us to log the messages from the nodes as the data is being received compared to non blocking send.

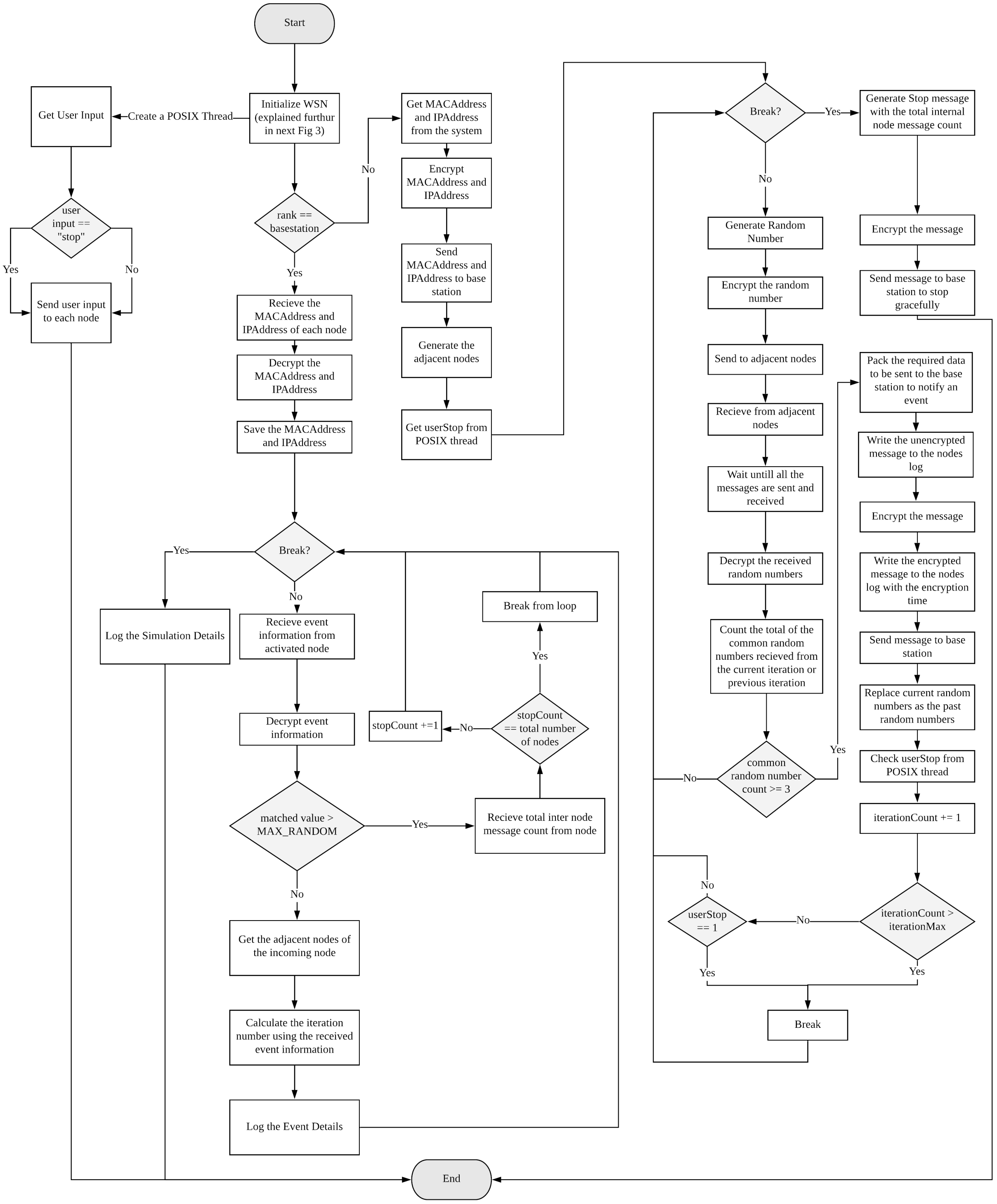

Start
Get User Input
Create a POSIX Thread
Initialize WSN (explained furthur in next Fig 3)
Get MACAddress and IPAddress from the system
user input == "stop"
Yes
No
Send user input to each node
rank == basestation
No
Yes
Encrypt MACAddress and IPAddress
Send MACAddress and IPAddress to base station
Generate the adjacent nodes
Get userStop from POSIX thread
Recieve the MACAddress and IPAddress of each node
Decrypt the MACAddress and IPAddress
Save the MACAddress and IPAddress
Break?
Yes
No
Generate Stop message with the total internal node message count
Encrypt the message
Send message to base station to stop gracefully
Generate Random Number
Encrypt the random number
Send to adjacent nodes
Recieve from adjacent nodes
Wait untill all the messages are sent and received
Decrypt the received random numbers
Count the total of the common random numbers recieved from the current iteration or previous iteration
common random number count >= 3
Yes
No
Pack the required data to be sent to the base station to notify an event
Write the unencrypted message to the nodes log
Encrypt the message
Write the encrypted message to the nodes log with the encryption time
Send message to base station
Replace current random numbers as the past random numbers
Check userStop from POSIX thread
iterationCount += 1
iterationCount > iterationMax
No
Yes
userStop == 1
No
Yes
Break
Break?
Yes
No
Log the Simulation Details
Recieve event information from activated node
Decrypt event information
matched value > MAX_RANDOM
Yes
No
Recieve total inter node message count from node
stopCount == total number of nodes
Yes
No
Break from loop
stopCount +=1
Get the adjacent nodes of the incoming node
Calculate the iteration number using the received event information
Log the Event Details
End

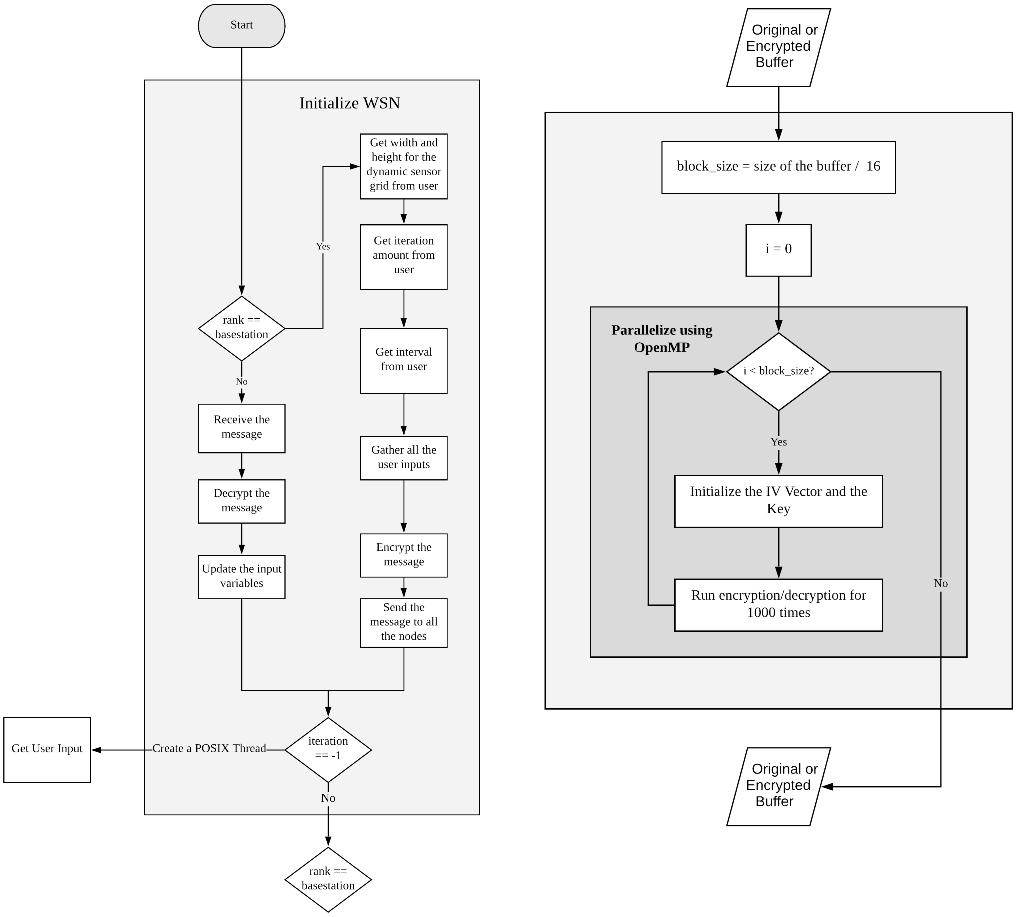
Fig. 2: Technical Flowchart for WSN Inter Process Communication

C. Encrypting communication and its effects

Due to the increase risk of cyber attacks, all wireless networks needs to be encrypted as its is easier to attack and steal unencypted data. So, all the messages are encrypted throughout the wireless sensor network. After comparing multiple encryption standards, Advanced Encryption Standard (AES) was used to encrypt the messages [12].

AES is a symmetric block cipher which can encrypt and decrypt information [14]. The Encryption part of the algorithm converts data into cipher text form while decryption part

Fig. 3: Technical Flowchart for sub parts of the system



(a) Technical Flowchart for Initializing the WSN

(b) Technical Flowchart for Encryption and Decryption algorithm with OpenMP

Start
Initialize WSN
rank == basestation
Yes
No
Get width and height for the dynamic sensor grid from user
Get iteration amount from user
Get interval from user
Gather all the user inputs
Encrypt the message
Send the message to all the nodes
Receive the message
Decrypt the message
Update the input variables
iteration == -1
Create a POSIX Thread
Get User Input
No
rank == basestation
Original or Encrypted Buffer
block_size = size of the buffer / 16
i = 0
Parallelize using OpenMP
i < block_size?
Yes
Initialize the IV Vector and the Key
Run encryption/decryption for 1000 times
No
Original or Encrypted Buffer

converts cipher text into text form of data. AES used different 128/192/256 bit keys to encrypt and decrypt data. For the WSN implementation we will be using AES 192, which uses a 192bit key. Since AES is the current standard for wireless communication, financial transactions and e-business we will be using it instead of other encryption algorithms [15]. As AES is a block cipher method, it has many modes of operations. Examples for these are Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Counter (CTR) [16].

For the WSN, we will be using the tiny-AES online implementation by kokke [17], which use the Counter (CTR) block cipher mode. In order to improve the speed of the encryption process we will be using OpenMP [4] to parallelize the process. This implementation uses the same method to encrypt and decrypt the buffer, so the method takes in the original message or the encrypted message. Then the original / encrypted message is broken into 16 byte chunks. Then each chunk is encrypted 1000 times. We use OpenMP with dynamic scheduling to parallelize the outer loop. The reason for using dynamic scheduling is explored and analyzed in Section IV-C of the report. The technical flow chart for the encryption and decryption algorithm is shown by Figure 3b. The speed up analysis for the encryption and decryption algorithm is discussed in Section IV-C.

III. METHODOLOGY

In this section we look into the methodology used to test and analyze both the event detection criteria as well as the encryption algorithm used in the WSN. Table I summaries the parameters used by the program to get the data required for analysis. In order to make the encryption and decryption time consistent, the buffer size for all the communication is constant. So the single random number communication between the sensor nodes uses this buffer size. This is inefficient, but allows us to accurately record the encryption and decryption times for all the messages in the network.

In order to analyze the encryption speed up, the three simulations are executed. One without using OpenMP, and the other two with static and dynamic scheduling of OpenMP [18].

All of the test cases were run on an Intel® Xeon® W-2145 Processor, which contains 8 cores and 16 logical threads. Figure 10 in the Appendix shows the screen shot of the console while testing. Another approach for the test case would be to use the dynamic stopping approach. Figure 11 in the Appendix shows the screen shot of the console for this approach.

For testing purposes, the time taken for encryption is logged by each node into a csv file and the other required data is also logged into a csv file from the base station.

TABLE I: Specifications of the system and the parameters used for generating the data to be analyzed

Specification	**Value**	**Description**
CPU	8/16	Number of CPUs and logical cores
RAM	64	Memory of the system
Parameter	**Value**	**Description**
Width	4	The width of the sensor node grid
Height	5	The height of the sensor node grid
Iterations	100	Total number of iteration performed by the system
Interval	1	Wait time between each iteration
MaxRandom	12	Limit the random numbers generated to between 0 - 11
Packsize	256	Size of the buffer for sending and receiving data

IV. RESULTS AND DISCUSSION

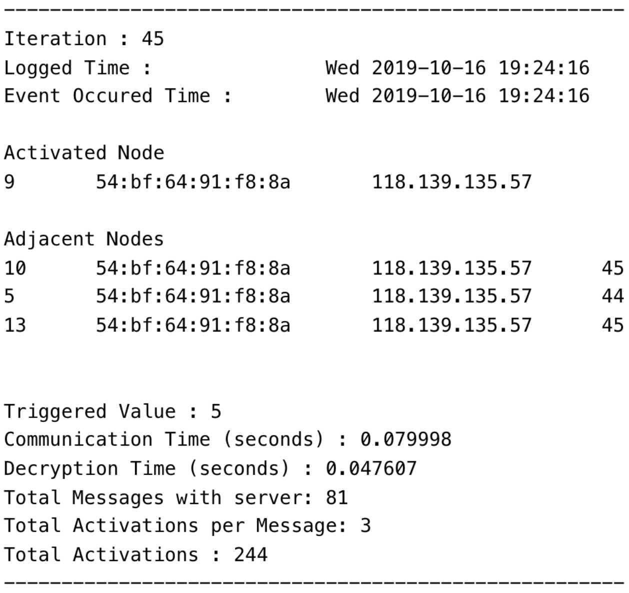
This section analyzes the results obtained using the methodology described in Section III. The analysis is broken down into three main sub sections, each explaining the different aspect of the result. First the structure of the log files are explained and the data of the log files are analyzed. Then the communication time is analyzed. Finally the encryption algorithm and the speed up of using OpenMP is analyzed.

A. Summary of Events

When an event is triggered by the sensor, the system generates two types of log files.

- Base Station generates the main log file which logs the event information and the simulation information
- Each sensor node logs the the original message, encrypted message and the encryption time

For readability purpose, parts of the log files will only be displayed in the report. The full log files and the testing files are included with the submission for the validity of the tests.



```
------------------------------------------------------
Iteration : 45
Logged Time :                Wed 2019-10-16 19:24:16
Event Occured Time :         Wed 2019-10-16 19:24:16

Activated Node
9       54:bf:64:91:f8:8a       118.139.135.57

Adjacent Nodes
10      54:bf:64:91:f8:8a       118.139.135.57      45
5       54:bf:64:91:f8:8a       118.139.135.57      44
13      54:bf:64:91:f8:8a       118.139.135.57      45


Triggered Value : 5
Communication Time (seconds) : 0.079998
Decryption Time (seconds) : 0.047607
Total Messages with server: 81
Total Activations per Message: 3
Total Activations : 244
------------------------------------------------------
```

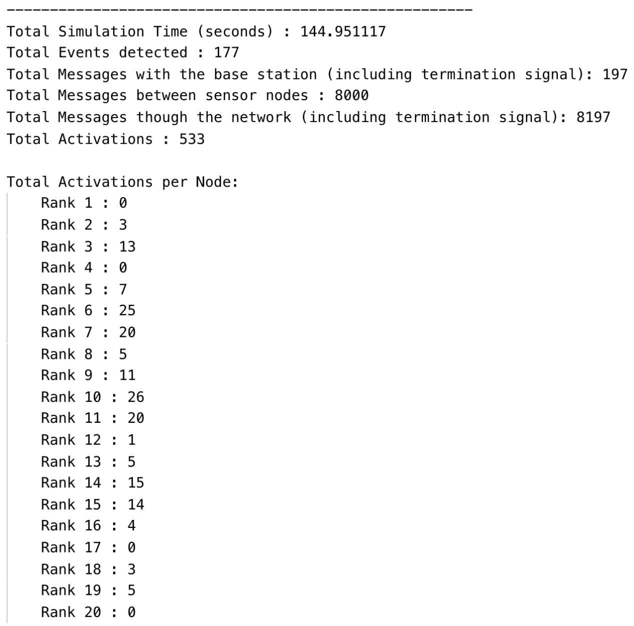
Fig. 4: Base station log file for event details

1) Base Station Log File: The base station logs two types of information, the event detection details and the simulation details at the end. First lets look into the log files to understand each part of it.

Figure 4 shows an event triggered at the 45 iteration of a simulation. This can be observed from the first line of the log which determines the iteration number. The next two lines of the log displays the time when the event was logged on the file and the actual time when the event is detected respectively. This is useful when the node and base station is present in two geographical locations and the time to send the message effects the logging time. But in this example, since the node is a process inside the same machine, we are not able to see a difference. The next two lines denote the activated node that was triggered i.e. the node that sent the message to the base station. The Rank, MAC Address and IP Address of the activated node is being logged in the log file. In a case where each node were running on separate machines, we are able to see different MAC and IP addresses. Next the adjacent nodes which generated the same random number is being logged. Similar to the activated node the Rank, MAC Address and IP Address of the adjacent nodes is also being logged in the log file. But there is also a fourth item that stores the iteration number. Since the WSN takes into account the random values generated from the previous iteration, this value helps to identify in which iteration the same random number was generated.

The next part of the log file shows the random number that triggered the event. Then the communication time and the decryption time is logged. The total messages with the base station shows how many messages are being sent to the server up until the current iteration. The total activations per message shows how many nodes are triggered for the current message. This can be either 3 or 4. Then finally the total activations up until the current iteration is also logged.

Figure 5 displays the log file that is generated at the end, when the system shuts down gracefully. It records the total simulation time, total number of events detected, total messages to the base station, total sensor node to sensor node messages and the total messages through the network. The total messages through the network contains the messages sent between the sensor nodes, messages sent from node to base station when an event is triggered and finally the messages sent from the sensor nodes with the termination signal to stop gracefully. It then logs the total activations through out the network and how many times each node was activated during the simulation according to the rank.



```
----------------------------------------------------
Total Simulation Time (seconds) : 144.951117
Total Events detected : 177
Total Messages with the base station (including termination signal): 197
Total Messages between sensor nodes : 8000
Total Messages though the network (including termination signal): 8197
Total Activations : 533

Total Activations per Node:
    Rank 1 : 0
    Rank 2 : 3
    Rank 3 : 13
    Rank 4 : 0
    Rank 5 : 7
    Rank 6 : 25
    Rank 7 : 20
    Rank 8 : 5
    Rank 9 : 11
    Rank 10 : 26
    Rank 11 : 20
    Rank 12 : 1
    Rank 13 : 5
    Rank 14 : 15
    Rank 15 : 14
    Rank 16 : 4
    Rank 17 : 0
    Rank 18 : 3
    Rank 19 : 5
    Rank 20 : 0
```

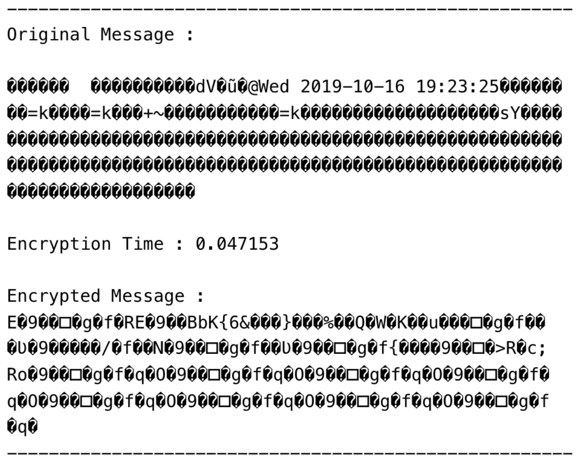
Fig. 5: Base station log file for simulation details

2) Sensor Node Log File: Next we look into the log file of each sensor node. Figure 6 shows the first event of the log file generated by the node with rank 3. This log file contains the original un-encrypted message, the encryption time and the encrypted message. If we analyze the log file we can see the time stamp in the original message but, after encrypting we do not see it.

B. Inter Process Communication Analysis

In this section we analyze the WSN data collected from the simulation.

1) Event detection messages from sensor nodes to base station: For each event detection, this architecture sends one message from the activated node to the base station. So referring to Figure 5, the total number of events generated is 177 and the total number of messages received from the nodes to the server is 197 which includes the 20 termination signals sent at the end. Figure 7a plots the number of messages received from the nodes over iterations. 7 out of the 20 nodes were activated at iteration 70, which is the highest amount of nodes activated per iteration. We can also see that after every iteration that has a high number of messages, the number of messages plummets down. The reason for this is, commonly high number of messages occur when there are matching random numbers in both the current iteration and the previous iteration. So after a high message iteration the previous array mostly contains 2 or more of the same common random numbers. So the probability of the random number of the current iteration being the same is low. Therefore the number of events detected in the next iteration goes down.



```
----------------------------------------------------
Original Message :

������  ����������dV�ũ�@Wed 2019-10-16 19:23:25������
��=k����=k���+~������������=k�������������������sY����
�����������������������������������������������������
�����������������������������������������������������
������������������

Encryption Time : 0.047153

Encrypted Message :
E�9��□�g�f�RE�9��BbK{6&���}���%��Q�W�K��u���□�g�f��
�Ʋ�9������/�f��N�9��□�g�f��Ʋ�9��□�g�f{����9��□�>R�c;
Ro�9��□�g�f�q�O�9��□�g�f�q�O�9��□�g�f�q�O�9��□�g�f�
q�O�9��□�g�f�q�O�9��□�g�f�q�O�9��□�g�f�q�O�9��□�g�f
�q�
----------------------------------------------------
```

Fig. 6: Sensor Node log file

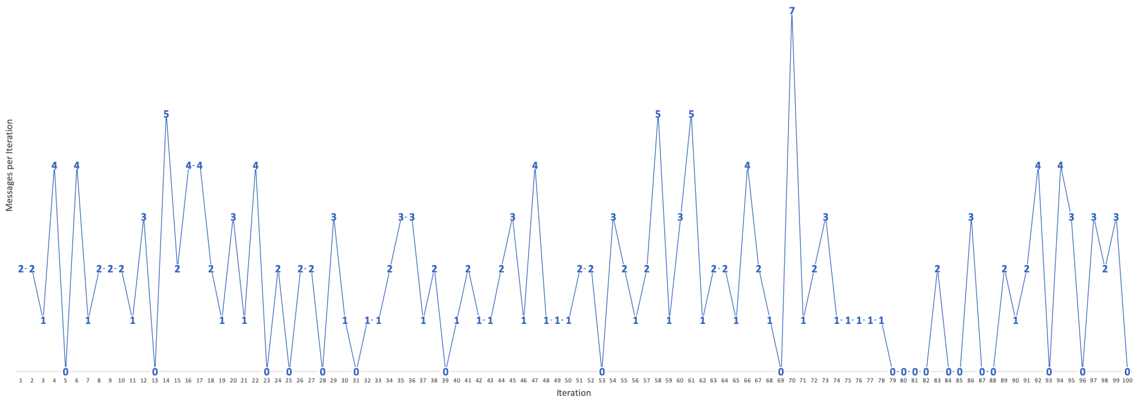
2) Total Events detected from each node: Next lets look at Figure 8, which is a heat map of the nodes and the events detected from each of them. The sensor nodes with four adjacent neighbors have the highest number of events generated as it has more probability of having 3 or more common random numbers. Also note the four corner nodes never detect an event as they only have 2 adjacent nodes. Therefore, the observations satisfy the actual probability of the event detection where the center nodes detect more events.

3) Communication time: The communication time includes the time for encrypting the message, sending the message, receiving the message and decrypting the message. Since the simulations are run on a single computer, the communication time from node to node must be similar in theory. But using Figure 7b we see that the times are different. The reason for this can be obvious if we compare Figure 7a and 7b. As a high number of events are detected close to each other, the communication time increases. This is due to the blocking communication used between the nodes and base station. But the red line in Figure 7a shows the tread line of the communication time. So even though there are peaks in the average communication time, the trends shows a constant speed as is should be in theory. Therefore, we can confirm that our assumption of constant time is true.

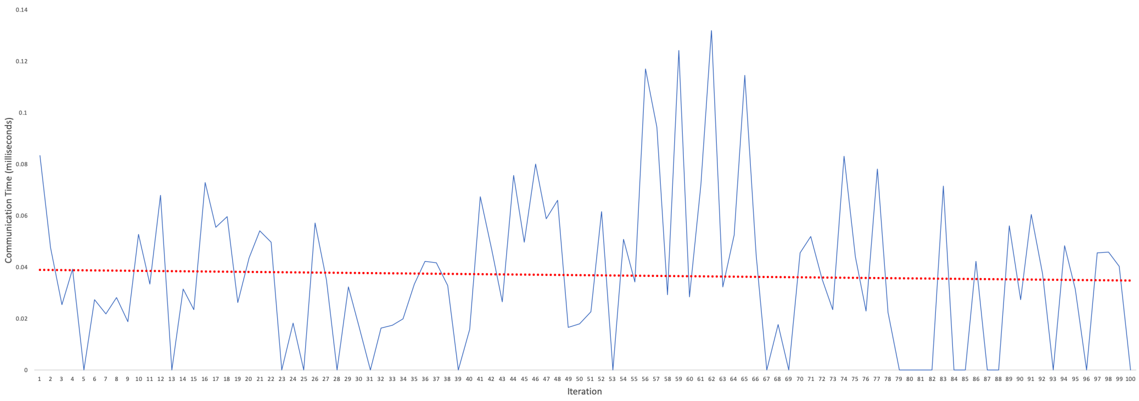
Another issue we might face is that when one part of the network is in a separate geographical location to the base station, the communication time will increase. So the nodes with high communication time will take longer to finish the

Fig. 7: Analysis of the data generated from the WSN

(a) Total Messages per Iteration



(b) Average Communication Time per Iteration



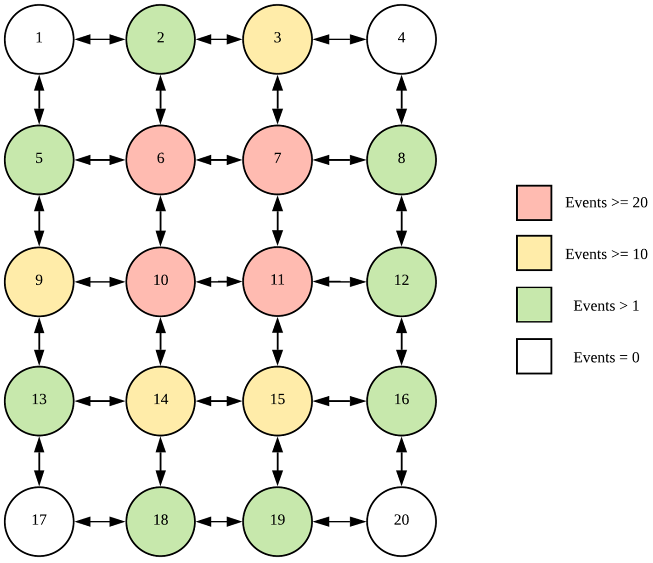


Fig. 8: Sensor Grid overlayed with the total event detection from each node

iterations. So the base station will wait until all the nodes have sent the termination signal to avoid any issue.

C. *Encryption Analysis*

In this section we will analyze how OpenMP helps improve encryption and decryption time of the AES algorithm.

In order to compare the speed up, first the encryption and decryption time of the AES algorithm without OpenMP is recorded. The encryption time is extracted from the node log files, where as the decryption time is extracted from the base station log file.

OpenMP has a static and dynamic way to parallelize a for loop [18]. In order to determine which way is better, simulations are run for each method and the speed is is calculated. This is done for both the encryption and decryption process.

We can calculate three types of speed up in the WSN.

Note : AT represents the average time, while SP represents the speed up.

- Speed up of Encryption which can be obtained by Equation 7.

$$SP_{encryption} = \frac{AT^{serial}_{encryption}}{AT^{parallel}_{encryption}} \tag{7}$$

- Speed up of Decryption which can be obtained by Equation 8.

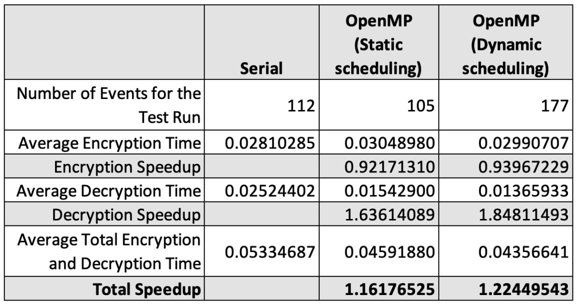
$$SP_{decryption} = \frac{AT^{serial}_{decryption}}{AT^{parallel}_{decryption}} \tag{8}$$

- Speed up of Encryption and Decryption which can be obtained by Equation 10. But in order to calculate this speed up we first get the average encryption time and decryption time, which is denoted by Equation 9.

$$AT_{total} = \frac{AT_{encryption} + AT_{decryption}}{2} \tag{9}$$

$$SP_{total} = \frac{AT^{serial}_{total}}{AT^{parallel}_{total}} \tag{10}$$

Figure 9, shows the results obtained by this experiment. We will now analyze the speed up based on two criteria.



	Serial	**OpenMP (Static scheduling)**	**OpenMP (Dynamic scheduling)**
Number of Events for the Test Run	112	105	177
Average Encryption Time	0.02810285	0.03048980	0.02990707
Encryption Speedup		0.92171310	0.93967229
Average Decryption Time	0.02524402	0.01542900	0.01365933
Decryption Speedup		1.63614089	1.84811493
Average Total Encryption and Decryption Time	0.05334687	0.04591880	0.04356641
Total Speedup		**1.16176525**	**1.22449543**

Fig. 9: Analysis of the OpenMP speed up for encryption and decryption

1) Encryption, Decryption and Total Speed up: If we analyze the results we see that the speed up for encryption time is lower than 1, but the decryption time and the total speed up is more than 1. The reason for this is because we are using a single computer for running the WSN. So each time multiple events are detected through out the network, the threads that are used for encryption by each node, floods the system. So on average the speed will be lower. but if we consider decryption, we can see a better speed up, as only one node i.e. base station creates the threads. But if we consider the total speed up for encryption, which includes both encryption and decryption, the speed up is higher than 1. This can be improved further by running each sensor node on different processor as it will have free threads to improve the time.

2) Static vs Dynamic Scheduling: Looking at the results, dynamic scheduling is better than static scheduling. The main reason for this that some threads get prioritized by the system scheduler. So if its dynamic, the high priority threads are able to perform more cycles compared to the static approach, which assigns the low priority threads with the same workload. Therefore dynamic scheduling is more suitable for parallelizing the encryption and decryption process.

V. CONCLUSION

Taking into account the hypothesis made in Section I, and all the observations and results made through the simulations in Section IV, let us condense the core outcomes of this report.

Section II delved into the theoretical and algorithmic details of the grid based IPC architecture. A high level technical explanation of the AES encryption algorithm and the benefits of it was also discussed.

Then Section III proposed a testing methodology to observe the effects of the proposed architecture. Three main criteria were taken into account when discussing the results of the simulations. Section IV-A summarized the logging portion of the system and then Section IV-B and IV-C analyzed how the communication time and encryption time affects the overall performance in a grid based architecture.

Now let us epitomize the two main finding of this report. Firstly the communication time of a grid based architecture can be dependent on the number of events as the base station causes a bottle neck when writing to the log files in serial. Next the encryption algorithm brings in more security to the system with the cost of more power and time to send events from the sensor nodes to the base station. This can be improved by using OpenMP to implement thread level parallelism, but, this in turn will bring in more power and resource usage which are limited in a WSN.

Conclusively, we can state that the grid based architecture has the benefit of faster event detection, as there is a direct link from the sensor nodes to the base station. This in turn allows the sensors to fire only when there is an event, so they can be in a power efficient state. We can also conclude that the parallelized AES encryption algorithm is able to reduce the communication times and also provide more security for the wireless sensor networks.

VI. FUTURE WORK

Many future research paths are opened as a result of this report. One of the possible paths to explore is the effects of sensors nodes in multiple geographical locations and how it effects the communication time. The fixed buffer size assumption that was taken in order to observe the speed up of the encryption algorithm can be removed and the affects of dynamic message length can also be explored. Then the observation that was made in Section IV-B1 where the messages plummet down after a set of high activation messages can be proved formally by testing it in a controlled environment. Finally, the system can be implemented in a prototype environment to observe the energy graphs of each node and analyze how the events and encryption effects the power consumption of the system.

Therefore, considering the aforementioned proposed future paths, we can state that this report paves the way to many exciting research areas on how an efficient Grid based Inter Process Communication Architecture affects Wireless Sensor Networks.

REFERENCES

[1] M. F. Othman and K. Shazali, “Wireless sensor network applications: A study in environment monitoring system,” *Procedia Engineering*, vol. 41, pp. 1204–1210, 2012.

[2] R. J. Barton, R. S. Wagner, and P. W. Fink, “Space applications of low-power active wireless sensor networks and passive rfid tags,” in *Wireless Sensor and Mobile Ad-Hoc Networks*. Springer, 2015, pp. 97–127.

[3] “Home,” Nov 2018. [Online]. Available: https://www.openmp.org/

[4] OpenMP, “Openmp,” Nov 2018. [Online]. Available: https://www.openmp.org/

[5] C. Guy, “Wireless sensor networks,” in *Sixth International Symposium on Instrumentation and Control Technology: Signal Analysis, Measurement Theory, Photo-Electronic Technology, and Artificial Intelligence*, vol. 6357. International Society for Optics and Photonics, 2006, p. 63571I.

[6] Techopedia.com, “What is inter process communication (ipc)? - definition from techopedia.” [Online]. Available: https://www.techopedia.com/definition/3818/inter-process-communication-ipc

[7] A. Onsman, “What is interprocess communication?” Sep 2018. [Online]. Available: https://www.tutorialspoint.com/what-is-interprocess-communication

[8] R. Miller, Q. F. Stout, and Q. F. Stout, *Parallel algorithms for regular architectures: meshes and pyramids*. Mit Press, 1996.

[9] X. Yang, H. Chen, S. Yang, L. Yi-Bing, and D. Ding-Zhu, “Wireless network security,” *EURASIP Journal on Wireless Communications and Networking*, 2009, copyright - Copyright © 2009 Yang Xiao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited; Last updated - 2012-01-28. [Online]. Available: https://search-proquest-com.ezproxy.lib.monash.edu.au/docview/856053027?accountid=12528

[10] T. Hoefler, R. Rabenseifner, H. Ritzdorf, B. R. de Supinski, R. Thakur, and J. L. Träff, “The scalable process topology interface of mpi 2.2,” *Concurrency and Computation: Practice and Experience*, vol. 23, no. 4, pp. 293–310, 2011.

[11] T. Hoefler and T. Schneider, “Optimization principles for collective neighborhood communications,” in *SC’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2012, pp. 1–10.

[12] J. Daemen and V. Rijmen, “Aes proposal: Rijndael,” 1999.

[13] U. Drepper and I. Molnar, “The native posix thread library for linux,” *White Paper, Red Hat Inc*, 2003.

[14] M. Rouse and M. Cobb, “What is advanced encryption standard (aes)? - definition from whatis.com.” [Online]. Available: https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard

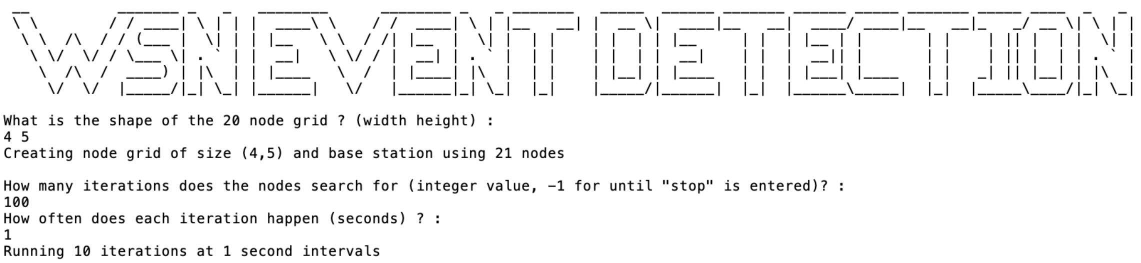
[15] R. W. World, “Advantages of aes.” [Online]. Available: https://www.rfwireless-world.com/Terminology/Advantages-and-disadvantages-of-AES.html

[16] K. Meghna, “Block cipher modes of operation,” Aug 2019. [Online]. Available: https://www.geeksforgeeks.org/block-cipher-modes-of-operation/

[17] Kokke, “kokke/tiny-aes-c,” Feb 2019. [Online]. Available: https://github.com/kokke/tiny-AES-c

[18] J. Corner, “Openmp: For scheduling,” Jun 2016. [Online]. Available: http://jakascorner.com/blog/2016/06/omp-for-scheduling.html

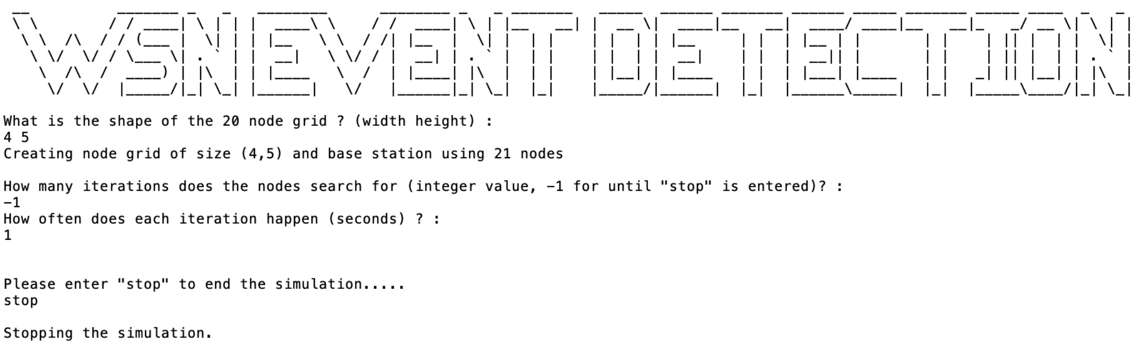
APPENDIX



```
What is the shape of the 20 node grid ? (width height) :
4 5
Creating node grid of size (4,5) and base station using 21 nodes

How many iterations does the nodes search for (integer value, -1 for until "stop" is entered)? :
100
How often does each iteration happen (seconds) ? :
1
Running 10 iterations at 1 second intervals
```

Fig. 10: Screenshot of the console used for generating of the results



```
What is the shape of the 20 node grid ? (width height) :
4 5
Creating node grid of size (4,5) and base station using 21 nodes

How many iterations does the nodes search for (integer value, -1 for until "stop" is entered)? :
-1
How often does each iteration happen (seconds) ? :
1


Please enter "stop" to end the simulation.....
stop

Stopping the simulation.
```

Fig. 11: Screenshot of the alternative approach to run the program until user enters stop