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

Samarasekara Vitharana Gamage, Bhanuka Manesha
bsam0002@student.monash.edu
28993373
School of Information Technology
Monash University
Malaysia

***Abstract*—**

***Index Terms*—**

I. INTRODUCTION

The rapid growth of 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 [2]. Currently Wireless Sensor Network (WSN) architectures are the standard for low power sensor networks. To accomplish low power usage each node in the WSN can only communicate with the four adjacent nodes and the base station. 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 [3]. Pipes, Socket, File, Signal, Shared Memory and Messege Queue are few of the different IPC architectures currently available [4]. 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 node i.e. nearest neighbors, a grid based socket IPC architecture would be the best method for the WSN.

Another constraint of WSNs, is that not only the have to operate on low power, but also be secure enough to broadcast sensitive data wirelessly. This is a major concern as most of the communication is done wirelessly and attackers can get the data easily [5]. To achieve this efficient encryption standards are used by the system.

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

II. THEORETICAL ANALYSIS AND INTER PROCESS COMMUNICATION DESIGN

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

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 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 is 4 and height is 5, while Figure 1b illustrates when its 2 and 10 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 height and width 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. *Do note that we first minus 1 from the current rank, this is due to the base station being rank 0.*

$$rowindex = (rank - 1) \,//\, width \tag{1}$$

$$columnindex = (rank - 1) \ mod \ width \tag{2}$$

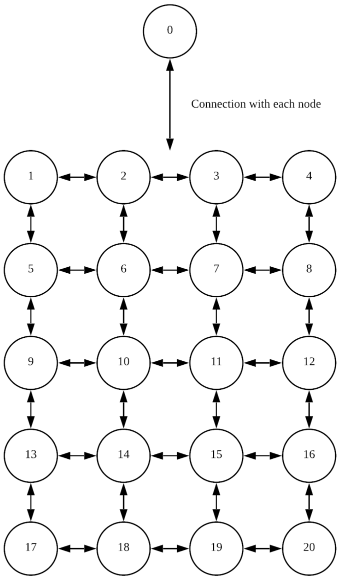
After determining the row index and the column index, we can use Equations 3, 4, 5 and 6 are used to get the four adjacent nodes.

$$left = rowindex \times width + columnindex \tag{3}$$

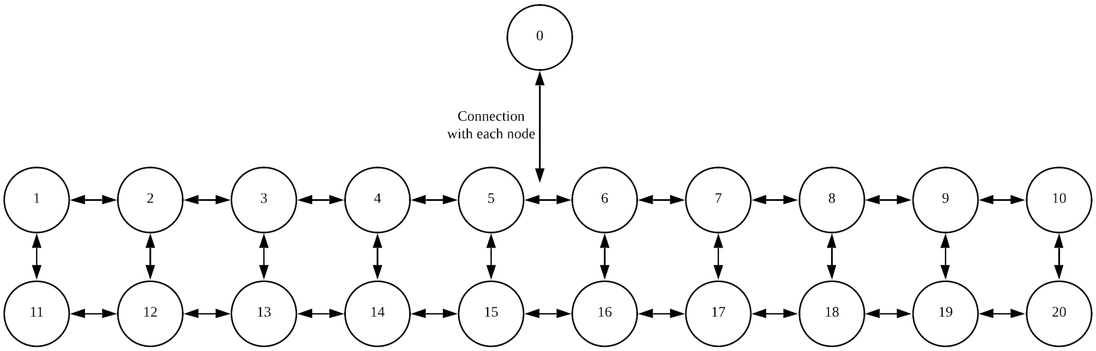
$$right = rowindex \times width + columnindex + 2 \tag{4}$$

Fig. 1: Dynamic Grid Layout

(a) 4 X 5 grid



(b) 10 X 2 grid



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

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

Note that the shift in the ranks due to base station is 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.

Published papers..

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 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 discussed in section II-C. The base station will then decrypt the message and store it in a dynamic array. Below is the data

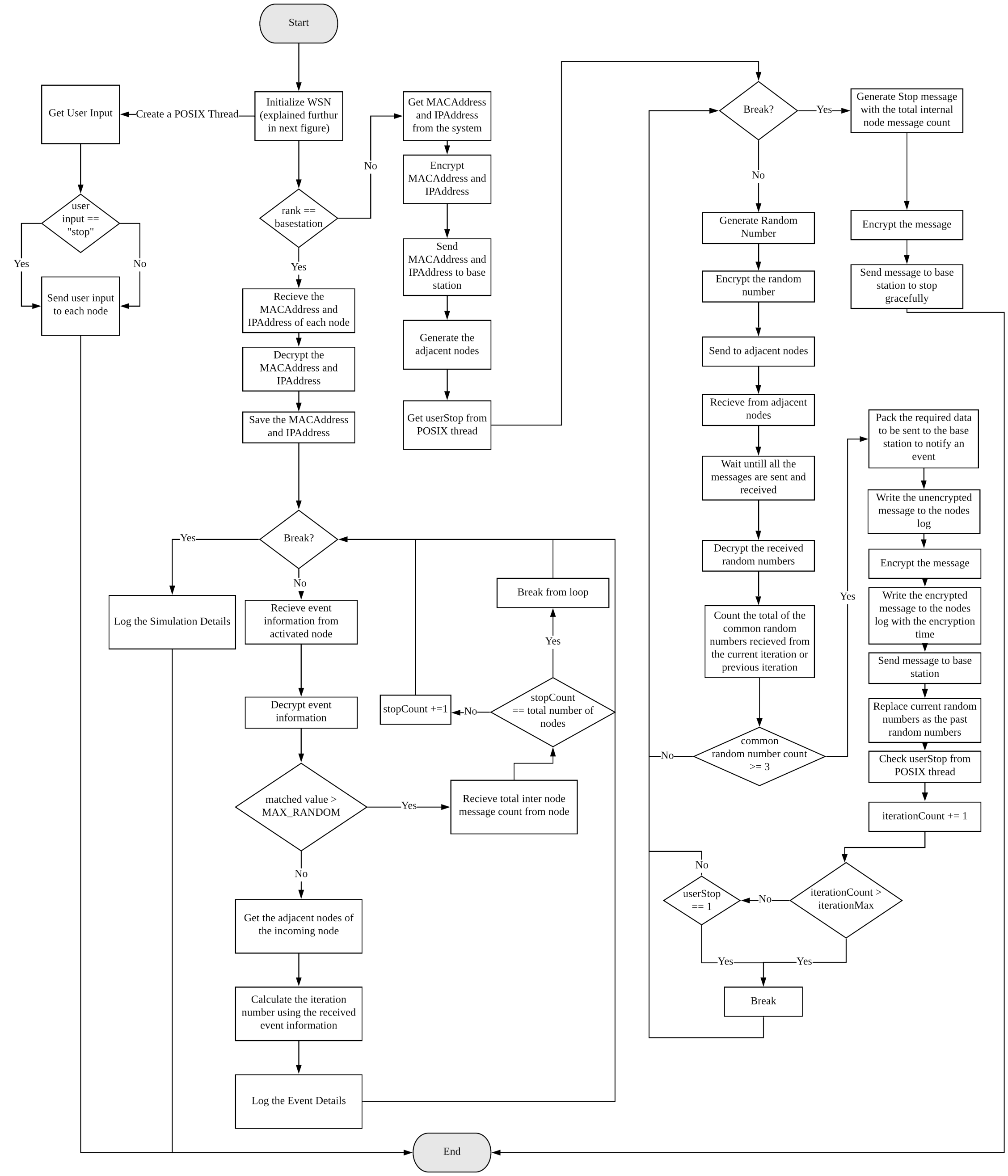

Start
Get User Input
Create a POSIX Thread
Initialize WSN (explained furthur in next figure)
Get MACAddress and IPAddress from the system
Break?
Yes
Generate Stop message with the total internal node message count
No
Encrypt MACAddress and IPAddress
user input == "stop"
rank == basestation
Yes
No
Yes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

Fig. 2: Technical Flowchart for WSN Inter Process Communication

passed from the node to the base station when initializing the system.

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

Now let us look into the algorithms used by the sensor nodes and the base station to generate and detect events.

1) Sensor Nodes: Firstly, 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

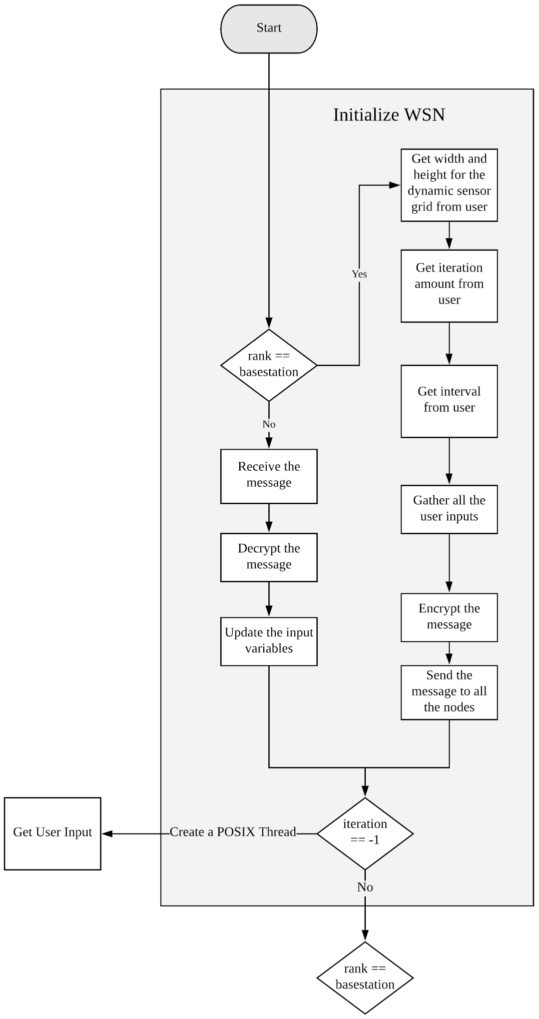

Start
Initialize WSN
Get width and height for the dynamic sensor grid from user
Yes
Get iteration amount from user
rank == basestation
Get interval from user
No
Receive the message
Gather all the user inputs
Decrypt the message
Encrypt the message
Update the input variables
Send the message to all the nodes
Get User Input
Create a POSIX Thread
iteration == -1
No
rank == basestation

Fig. 3: Technical Flowchart for Initializing the WSN

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 users stop signal is received, the node will send a termination signla to the base station to exit out of the system gracefully.

2) 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 user enters the keyword ”stop”. In order for this to work, the base station creates a POSIX thread, which enables it to check for user inputs as well log the events from the nodes. When the user inputs the keyword ”stop”, 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.

Figure 2 breaks down the algorithm of the whole system in to a technical flowchart. Then Figure 3 explains how the system initializes the WSN, based on the dynamic user inputs.

Blocking and Non Blocking Send

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.

AES is a symmetric block cipher which can encrypt and decrypt information [6].The Encryption part of the algorithm converts data into cipher text form while decryption part 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 [**?**]. Block Cipher has many modes of operations. Since AES is also a block cipher, it has many modes of operations. Examples for these are Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Counter (CTR) [7].

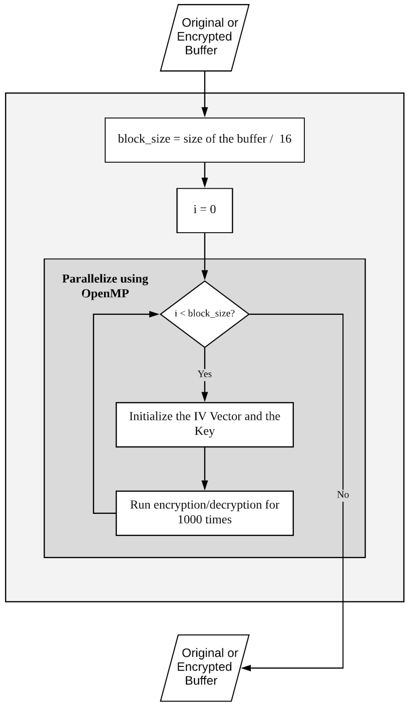

Original or Encrypted Buffer
block_size = size of the buffer / 16
i = 0
Parallelize using OpenMP
i < block_size?
Yes
Initialize the IV Vector and the Key
Run encryption/decryption for 1000 times
No
Original or Encrypted Buffer

Fig. 4: Encryption and Decryption algorithm with OpenMP

For the WSN, we will be using the tiny-AES online implementation by kokke [8], which use the Counter (CTR) block cipher mode. In order to improve the speed of the encryption process we will be using OpenMP [9] to parallelize the process. This implementation uses the same method to encrypt and decrypt the buffer. Since the AES implementation encrypt and decrypt a block size of 16 bytes for 10 rounds, OpenMP is applied on the inner loop. So the 10 rounds are done in parallel. Figure 4 attempts to explain this. 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 a constant size. So even the single random number communication uses a buffer. This is inefficient, but allows us to accurately record the encryption and decryption times for all the messages in the network.

All of the test cases were run on a 2016 MacBook Pro with an Intel 6820HQ processor, which contains 4 cores and 8 logical cores. Figure 11 in the Appendix shows the screen shot of the console. Another approach for the test case would be to use the dynamic stopping approach. Figure 12 in the Appendix shows the screen shot of the console for this approach.

TABLE I: Parameters used for generating the data to be analyzed

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 then the data of the log files are 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 are inside the submission folder.

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 file to understand each part of it.

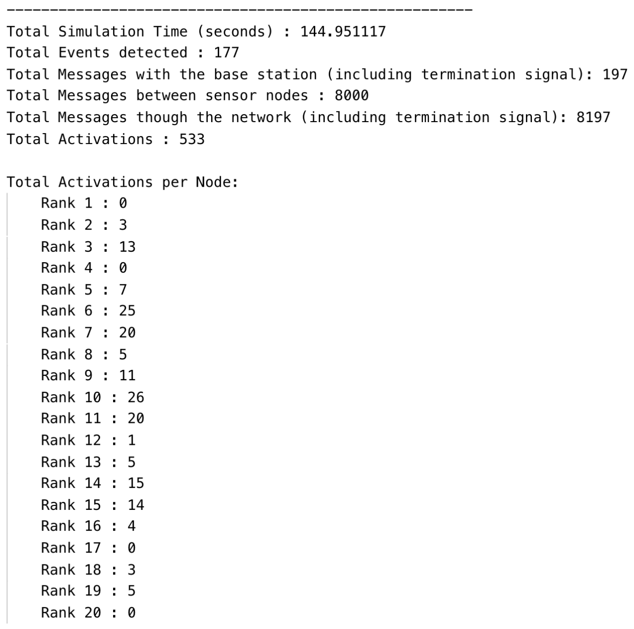
```
------------------------------------------------------
Iteration : 45
Logged Time :                Wed 2019-10-16 19:24:16
Event Occured Time :         Wed 2019-10-16 19:24:16

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. 5: Base station log file for event details

Listing 5 shows an event triggered at the 98 iteration of our test. This can be observed from the first line of the log which determines the iteration number. The next two lines of the logs 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 adjacent node the Rank, MAC Address and IP Address of the activated node 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 help to identify which 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.

Listing 6 displays the log file that is generated at the end, when the system turns off gracefully. It records the total simulation time, 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 also logs the total activations though out the network and how many times each node was activated during the simulation.



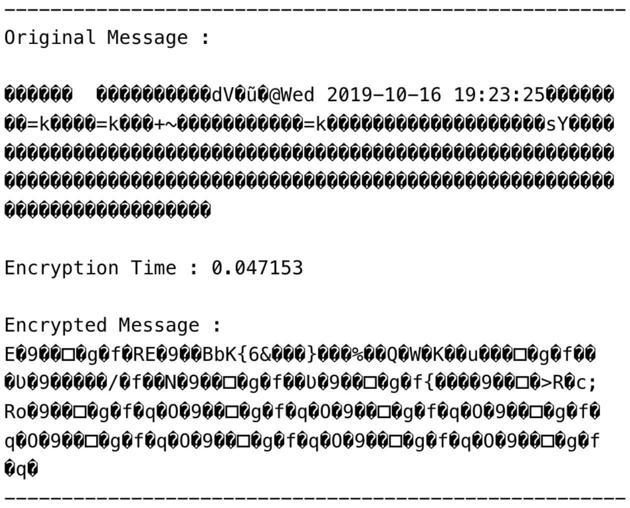
```
----------------------------------------------------
Total Simulation Time (seconds) : 144.951117
Total Events detected : 177
Total Messages with the base station (including termination signal): 197
Total Messages between sensor nodes : 8000
Total Messages though the network (including termination 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. 6: Base station log file for simulation details

2) Sensor Node Log File: Next we look into the log file of each sensor node. Figure 7 shows the first event of the log file generated by the node with rank 6. 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. Do note the encryption time shown in Figure 7 is not the encryption time used for the speed up analysis.

B. Inter Process Communication Analysis

In this section we analyze the WSN data collected from the test case. We will explore the following sections of the data in this section.



```
----------------------------------------------------
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. 7: Sensor Node log file

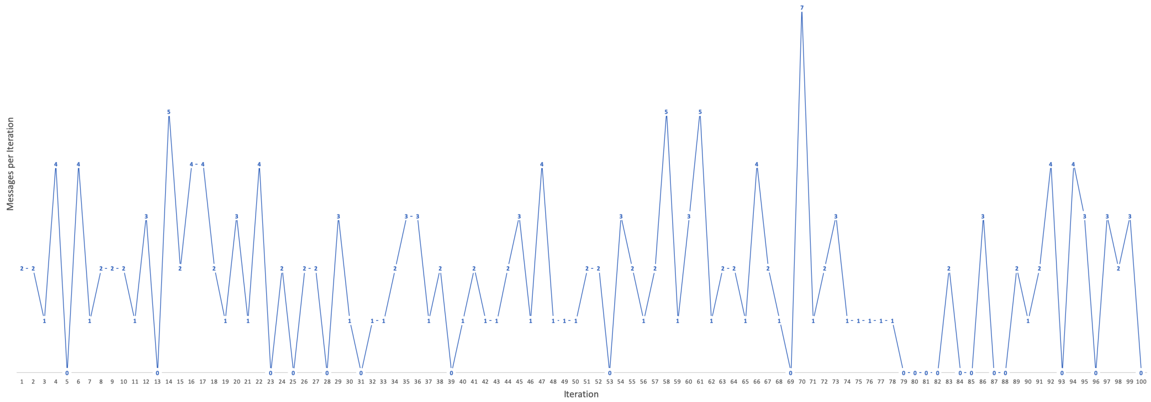
1) Event detection messages from sensor nodes to base station: For each event detection, this architecture sends one message from the activated node to the base station. So referring to listing 6, the total number of events generated is 459 and the total number of messages received from the nodes to the server is 479 which includes the 20 termination signals sent at the end. Figure 8a plots the number of messages received from the nodes over iterations. To highest number of messages of 13 out of the 20 nodes were received at iteration 79. 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 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. Now referring Figure **??**, we see that it also follows the same pattern as Figure 8a. But the counts is different as each message can have 3 or 4 activations.

2) Total Events detected from each node: Next lets look at Figure 9, 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 number. Also note the four corner node never detect an event as they only have 2 adjacent nodes. Therefore the observations satisfy the actual probability of the event detection.

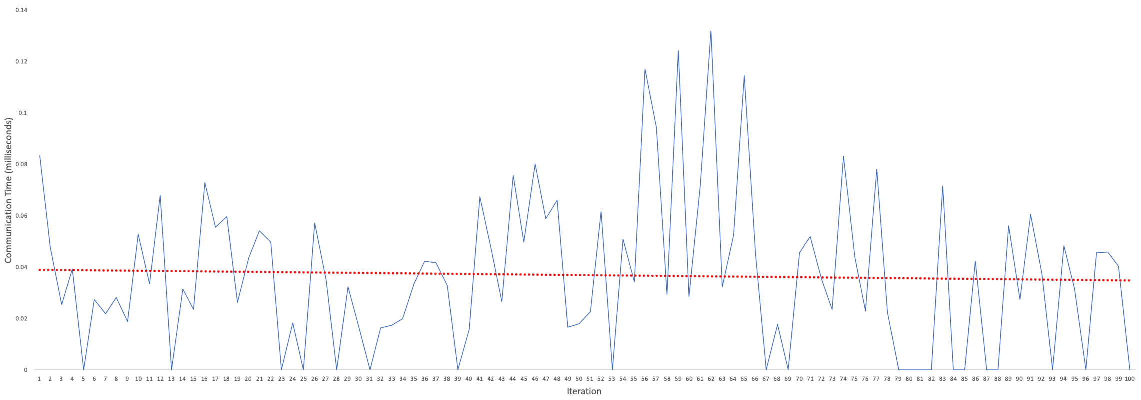
3) Communication time: The communication time includes the time for encrypting the message, sending the message, receiving the message and decrypting the message. Since the tests case is run on a single computer, the communication time from node to node is similar except between iteration 55-70. This can be observed by Figure 8b, where the is a peak in the

Fig. 8: Analysis of the data generated from the WSN

(a) Total Messages per Iteration



(b) Average Communication Time per Iteration



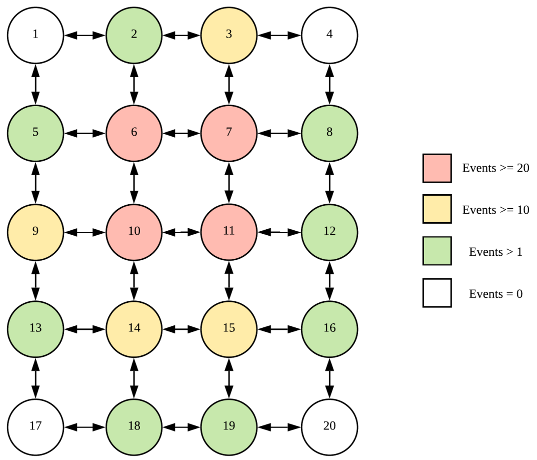


Fig. 9: Sensor Grid overlayed with the total event detection from each node

average communication time. But if one part of the network is in a separate geographical location to the base station, the communication time will increase. The issue that arises with this is that the nodes with high communication time will take longer to finish the iterations. So the base station will wait until all the nodes have sent the termination signal to avoid any issue. One way to improve the communication time is by parallelizing the encryption and decryption algorithms which we will look into in the next section.

C. Encryption Analysis

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

In this section we will analyze how OpenMP helps improve encryption and decryption time of the AES algorithm. Since we are using the CTR mode operation of the AES algorithm, the algorithm is parallelizable. The buffer size of the messages used for testing the encryption is different from the actual buffer size which was used to get the results. The reason for this is the smaller buffer size is insignificant to see any speed up when parallelizing the code as they are in microseconds. So for testing the speed up of the encryption and decryption we used a buffer size of 160000 initialized with zeros.

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 extracted from the base station log file.

OpenMP has a static and dynamic way to parallelizing a for loop **??**. In order to determine which way better, average of two runs from each method are obtained and the speed is is calculated. This is done for both the encryption and decryption process with a pack size of 160000. Figure **??** and **??**, shows the results obtained by the experiment and the static approach has a higher speed up for both encryption and decryption. The reason for this is that dynamic is better when the time taken for each iteration is not constant, but static is better when each iteration takes the same amount of time. Since each iteration of our loop is takes a constant time, the static approach is able to outperform the dynamic one. Therefore the WSN implementation uses the static approach.

V. CONCLUSION

VI. FUTURE WORK

Different grid architectures in different locations

Prove the drop of events

optimize the packsize to be dynamic based on the data sent

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] 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.

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

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

[5] 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

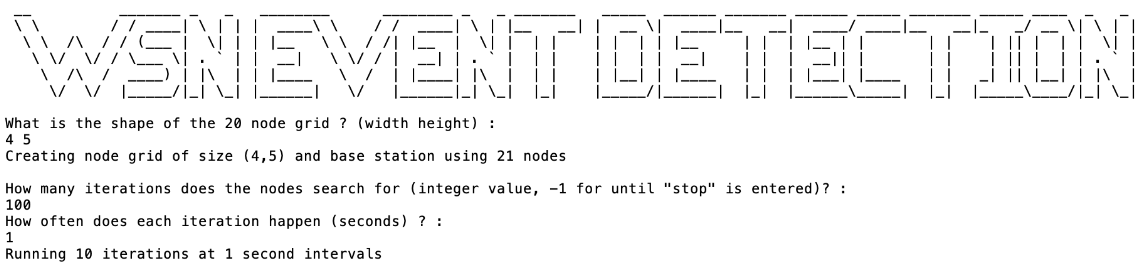
[6] M. Rouse and M. Cobb, “What is advanced encryption standard (aes)? - definition from whatis.com.” [Online]. Available: https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard

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

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

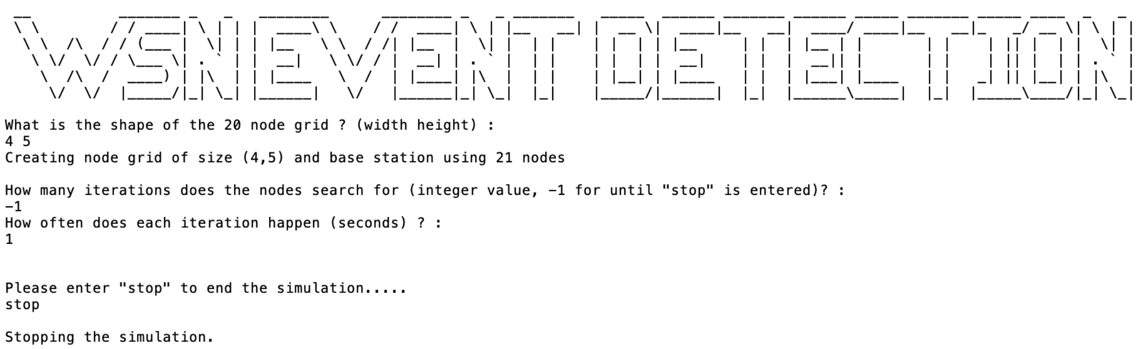
[9] OpenMP, “Openmp,” Nov 2018. [Online]. Available: https://www.openmp.org/

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. 11: Screenshot of the console used for the generating 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. 12: Screenshot of the alternative approach to run the program until user enters stop