Event Detection in a fully distrubuted wireless sensor network

James Schubach
Monash University
Melbourne, Australia
schubach.james@gmail.com

***Abstract*—The use of wireless sensors is extremely common in 2019 and as such are the problems involved with them. A common problem which we explore in this paper is the stress on the base station due to messages being sent from the sensors. Thus because of this the aim of the paper is to figure out if the use of nearest neighbor elevates some of the stress by reducing the messages being sent. To do this, an algorithm was created in C to simulate a WSN from here the sensors generated random numbers and then send them to its neighbors. If 3 of the neighbors had the same value, then a event was triggered and was sent to the base station. The results showed that the use of nearest neighbor had a significant impact on the events being sent off, this is due to the nature of the algorithm. However, after implementing OpenMP it did slow down the WSN, however this is due to limitations of the system, rather than the algorithm**

Keywords-component; Wireless Sensor Network, OpenMP, MPI, C, nearest neighbor, Distrubuted Computing

I. INTRODUCTION

The assignment was based on a grid of distributed wireless sensors, we were tasked with creating a solution to the problem, such that sensors can only talk to adjacent sensors. E.g. sensor 0 can talk to 1 and 4 and the base station. From here the sensors are to generate random values and then send them to the adjacent nodes. After this is done, they receive the values from those nodes, this is where the event detection arises. The sensor will then check to see if 3 or more adjacent nodes have the same random value, if so then a event is triggered, that node will then report to the base station notifying that an event has occurred. All messages to and from the base station along with adjacent nodes were to be encrypted by utilizing OpenMP. The objectives of the assignment were to simulate a WSN efficiently, by utilizing clock coordination, ring communication, encryption and IPC.

IPC refers to the instruments in place to enable processes to communicate with one another, this can include shared data [1]. By implementing a near-neighbor approach to this problem, it was hypothesized that this would reduce the overall messages being passed on to the base station. This also allowed the requirement of only being able to talk to adjacent nodes.

II. THEORETICAL ANALYSIS & IPC DESIGN

A. Technical Illustration (Refer to Figure 1)

The figure shows the communication scheme between the processes. As shown, 21 process are split into two groups the sensors of size 20 and the base station of size 1. From here, the sensors all have communication lines to the sensors to their direct North, East, South and West. This enforces the near-neighbor approach. The arrow pointing towards the base station shows that any process can communicate with it, however the sensor will only do so if they trigger an event. In the paper by Xie W et al. [2] report they found by using a similar algorithm K Nearest neighbors (which instead of being directly connected, it will be K far away), that this allowed the system to not be "flooding the message in the whole network". In the paper by Mohammed Ahmed et al. [3] they suggest that by implementing the K-Nearest neighbor algorithm reduces energy consumption, thus "prolong network lifetime". From these articles, it has been thoroughly researched the nearest neighbor's approach is not only efficient by also effective and minimizing message passing (by assuming less message passing means less energy consumption).

B. Encryption/Decryption Algorithim via OpenMP

The encryption algorithm used for this assignment is referred to as the 'Tiny Encryption Algorithm' created by David Wheeler and Roger Needham [4]. While this algorithm isn't as complicated as AES, it makes up for that in a large amount of iterations. The algorithm is based on Feistel routine. Stated in the paper, that the algorithm "… is of the same order of complexity of DES". As DES is a wide used algorithm for encryption, this small portable algorithm does the job for our assignment. The actual algorithm with use of OpenMP was taken from Jackson Goerner [5]. Goerner has implemented OpenMP in the main iteration cycle of the algorithm, thus reducing the time taken to encrypt/decrypt by the number of threads. We still want this system to be fast, so introducing more complex algorithms would slow down the sensor detection rates, thus making the WSN useless.

C. *Flowchart (Refer to Figure 2)*

As per the figure, the flow of the program is shown. The basis of the algorithm is as follows, set rank 20 to the base station and all other ranks to sensors. From here the sensors will constantly be looping through the phase of generating random numbers, receiving from other sensors. Checking for an event, sending event of it has detected. Then rank 0 being the coordinator will check to see if a second has passed since the beginning, if it has it notifies the sensor next to it and these repeats. They all will break out of the loop and start the next iteration. This allows the time to be synced up between the sensors. The base station is solely probing for messages, if it receives a message, it checks to see if it is an event message or exit message. If its and event message it will receive the event, decrypt it print it to the log file and then go back to waiting for a message. If it's an exit message, it will print the summary to the log file and then exit. The base station will only exit once all 15 iterations have been complete (or 15 seconds), rank 0 will then notify the base station that it needs to exit.

III. RESULTS

A. *Table of Results (Refer to Table 1)*

As per the table, you can see the communication time between the sensor and base station are under 10 seconds for the majority. This could be because of a backlog of events being processed, or more significantly due to the encryption/decryption time. However, communication time between two sensors was an insignificant amount. Once again, the any variances in time was from a small backlog, however due to using the near neighbor approach this significantly reduces the number of messages a node can receive. This also significantly reduce the number of messages the base station receives. Our system should run for 15 seconds, however as the number of events rack up, we see that the time taken for the simulation also increases. This is due to the encryption time; the average decryption time was 0.6 seconds and then if we say that the encryption time was similar it takes 1.2 seconds per encrypt/decrypt. This would account for the variances in time between time taken and theoretical time taken. This shows via our hypothesis that using nearest neighbor significantly speeds up our time, rather than every node reporting to the base station. Also, the time taken to decrypt/encrypt was recorded on a 8-core CPU, as that was already split by 21, the use of OpenMP has no significant benefit to the speed-up. If the simulation were to run on a HPC we would see a significant speed up in time and thus reducing the over-all time taken.

IV. CONCLUSION

From the report, it shows a clear understanding that the use of nearest neighbor has not only been thoroughly researched it has also been tested. The findings of the research and testing have found that the use of nearest neighbor to reduce the load/ messages sent to the base station by a considerable amount and thus proves our hypothesis correct. It would be a good idea to test the simulation on a larger more powerful system, not only to increase the effectiveness of OpenMP regarding encrypting/decrypting, but also any overheads of splitting a processor up into smaller amounts.

REFERENCES

[1] "Inter-process communication," Wikipedia, 26-Sep-2019. [Online]. Available: https://en.wikipedia.org/wiki/Inter-process_communication.

[2] W. Xie, X. Li, V. Narasimhan, and A. Nayak, "K Nearest Neighbour Query Processing in Wireless Sensor and Robot Networks," Ad-hoc, Mobile, and Wireless Networks Lecture Notes in Computer Science, pp. 251–264, 2014.

[3] M. M. Ahmed, A. Taha, A. E. Hassanien, and E. Hassanien, "An Optimized K-Nearest Neighbor Algorithm for Extending Wireless Sensor Network Lifetime," The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018) Advances in Intelligent Systems and Computing, pp. 506–515, 2018.

[4] W. David, N. Roger, "TEA, a Tiny Encryption Algorithim," 1994

[5] G. Jackson, "ParallelEncryption/tea_omp.c," Github, 26-Sep-2019. [Online]. Available: https://github.com/glipR/ParallelEncryption/blob/master/tea_omp.c.

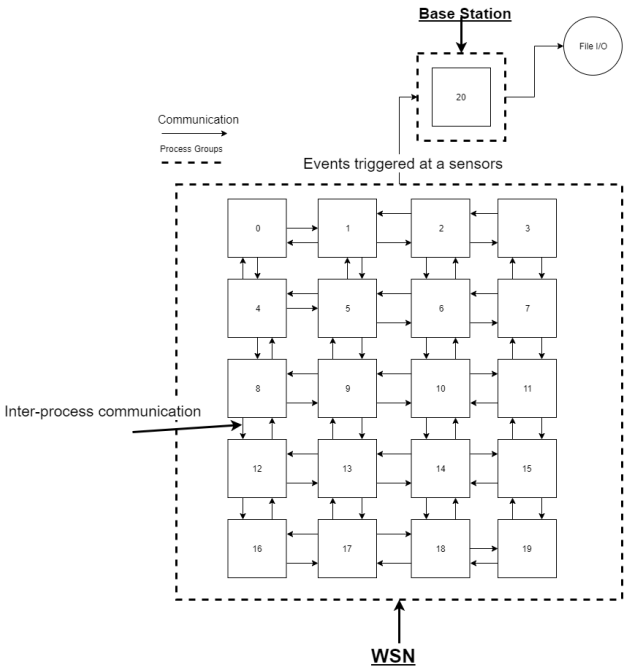

Figure 1. Detailed Illustraction of a near-neighbour IPC architecture.

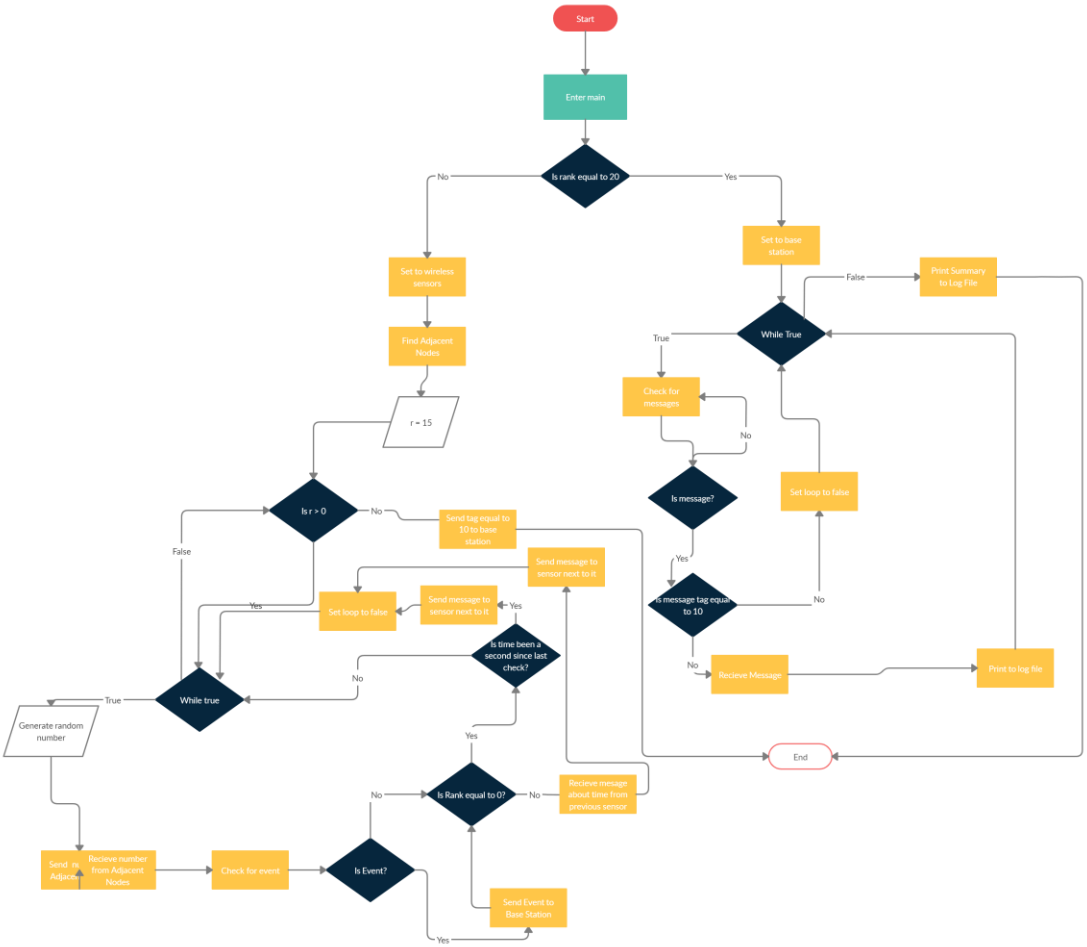

Figure 2. Detailed Flowchart of Algorithim.

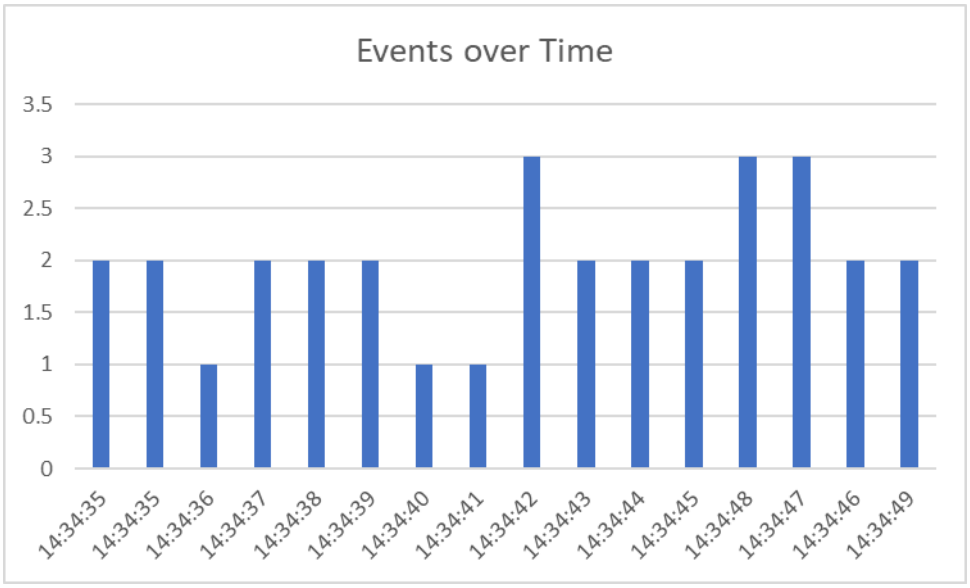

Figure 3. Chart of Events over time

Runs #	Reported Event	Summary of Events	
1	23	Summary: Simulation completed Sucessfully! Events Detected: 23 Time Taken in sec: 16.182287 Average Time for a Decrypt: 0.692469 Non-event Messages: 1	Time to decrypt: 0.449402 Event triggered at Sensor: 15 Nodes: 11, 14, Value is: 120 Network Address: 127.0.1.1 Time Sent: Sun Oct 20 14:16:52 2019 Time Recieved: Sun Oct 20 14:16:52 2019 Total Events so far: 23
2	36	Summary: Simulation completed Sucessfully! Events Detected: 36 Time Taken in sec: 21.951597 Average Time for a Decrypt: 0.600875 Non-event Messages: 1	Time to decrypt: 0.444843 Event triggered at Sensor: 13 Nodes: 17, 12, Value is: 273 Network Address: 127.0.1.1 Time Sent: Sun Oct 20 14:27:12 2019 Time Recieved: Sun Oct 20 14:27:19 2019 Total Events so far: 36
3	41	Summary: Simulation completed Sucessfully! Events Detected: 41 Time Taken in sec: 24.225205 Average Time for a Decrypt: 0.584302 Non-event Messages: 1	Time to decrypt: 0.441955 Event triggered at Sensor: 19 Nodes: 15, 18, Value is: 203 Network Address: 127.0.1.1 Time Sent: Sun Oct 20 14:30:15 2019 Time Recieved: Sun Oct 20 14:30:25 2019 Total Events so far: 41
4	34	Summary: Simulation completed Sucessfully! Events Detected: 34 Time Taken in sec: 21.075950 Average Time for a Decrypt: 0.613851 Non-event Messages: 1	Time to decrypt: 0.445084 Event triggered at Sensor: 12 Nodes: 8, 13, Value is: 226 Network Address: 127.0.1.1 Time Sent: Sun Oct 20 14:32:19 2019 Time Recieved: Sun Oct 20 14:32:27 2019 Total Events so far: 34
5	19	Summary: Simulation completed Sucessfully! Events Detected: 19 Time Taken in sec: 16.448074 Average Time for a Decrypt: 0.677395 Non-event Messages: 1	Time to decrypt: 0.445688 Event triggered at Sensor: 13 Nodes: 14, 12, Value is: 160 Network Address: 127.0.1.1 Time Sent: Sun Oct 20 14:33:27 2019 Time Recieved: Sun Oct 20 14:33:27 2019 Total Events so far: 19
6	30	Summary: Simulation completed Sucessfully! Events Detected: 30 Time Taken in sec: 19.264266 Average Time for a Decrypt: 0.628411 Non-event Messages: 1	Time to decrypt: 0.445275 Event triggered at Sensor: 14 Nodes: 18, 13, Value is: 141 Network Address: 127.0.1.1 Time Sent: Sun Oct 20 14:34:49 2019 Time Recieved: Sun Oct 20 14:34:54 2019 Total Events so far: 30
	Events Detected	Time Taken to Decrypt with OpenMP	
Averages	30.5	0.632883833	

Table 1. Details of simulated runs