Event detection in a fully distributed wireless sensor network

Author's name: Thuan Anh Bui
Faculty of Information Technology
FIT3143 Parallel Programming
Monash University, Clayton Campus
Melbourne, Australia
Email: tbui0003@student.monash.edu

Abstract – The purpose of the assignment is to simulate how wireless sensor network functions by setting up a virtual design and implementing this design containing a base station and its routers (depicted as nodes). The base station's main duty is to collect data from these routers and display output. The sensor node when triggered sends the encrypted data and the base station decrypt the data. The overall result for the communication time shows that the total time taken for the sensor nodes to send data to the base station vary.

Keywords – Event, sensor nodes, base station, adjacent nodes, message, encryption, decryption, communication time

I. INTRODUCTION

Inter-process communication describes the mechanism that one process communicates with another process to share some information or data. There are two approaches to this mechanism:

- Shared data approach
- Message passing approach

The former enables a common data memory storage where process 1 forwards its data over and process 2 extracts the data from the storage. The same thing is done by process 2 towards process 1. Both the 2 processes and the memory storage are stored in the stack where the latter is located between the 2 processes [1]. Nevertheless, this approach is not the focus of implementation in the report while the latter will be conducted.

The latter method allows direct communication between process 1 and process 2. The investigation conducted will involve 2 different types of communication:

- The communication between the base station and its slave nodes.
- The communication between among a node and its adjacent nodes.

The purpose of the investigation is to enable all the nodes with 3 or more adjacent nodes to send their data to the base station. In addition, if there are triggering nodes with same data, only the delegating node is responsible for sending the data. This design will accomplish 2 objectives which are to have the triggering event occurs when there are more than 3 adjacent nodes next to a reference node and to avoid too many nodes with same data sending to the base station.

In order to depict the synchronization of the networks, the IPC algorithm exploits the number of iterations with each iteration is tied to a sensor node's action. The following metrics will be used to evaluate the wireless sensor network among the nodes and the base station:

- The time taken to encrypt the message, forwarding it to the base station and decrypt it
- The total number of messages in each iteration
- The total number of events in each iteration
- A summary of the all the nodes happening in each iteration

Initial hypothesis on how the wireless sensor network is expected to operate:

- The encryption/decryption algorithm used to encrypt the message is Caesar cypher[2]. Due to its simplicity and the length of each message is relatively small, the time taken for Caesar cypher to encrypt and decrypt is less than 1 second.

TABLE 1

Message	1	2	3
Encryption time (in second)	0.064643	0.004178	0.001405
Decryption time (in second)	0.061767	0.010229	0.001086

- The number of events depends on how many adjacent nodes next to the sensor node.

TABLE 2

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

- The total number messages for each iteration is based on the formula:

$$
\begin{aligned}
&Number\ of\ messages\\
&= (Sensor\ sends\ to\ adjacents)\\
&+ (Adjacents\ send\ to\ sensor)\\
&+ (Sensor\ sends\ to\ base\ station)
\end{aligned}
$$

For instance, if there are 3 adjacent nodes, the first count is 3 as sensor signals its adjacent nodes. Each adjacent node forwards back its random number so the count is incremented to 6.
There are 2 scenarios in the third case. If all nodes have the same random number, the sensor only forwards to the base station the data once. Otherwise, if there is a difference in a node or two, the sensor needs to forward 2 or 3 times to the base station.

II. IPC ARCHITECTURAL DESIGN

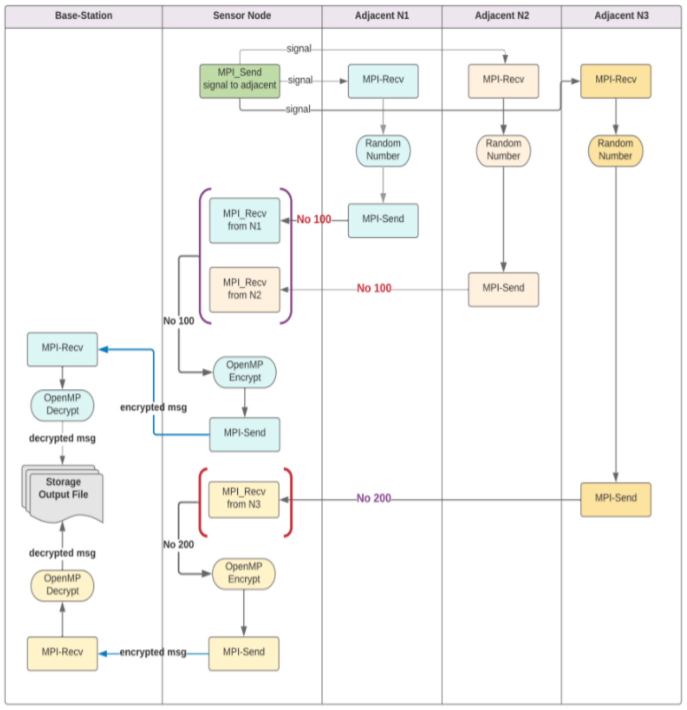
The common method used for the IPC design is to use Send and Receive in Message Passing Interface. In addition, by just having a random number as a message is too short to test the encryption algorithm. The random number occupies the data value of the message and the message will be 20-character long to ensure that the encryption is very randomized. The image 1 below implies the example of how the IPC architecture is used to demonstrate the functioning of the sensor wireless network.

At the initial stage of the IPC program, each node is checked to ensure that only nodes with 3 or more adjacent nodes are eligible for event detection. All eligible sensor nodes trigger a signal to these adjacent nodes. A generated random number is assigned to each of these nodes. This random number is a data number forming a part of a message; therefore, the content of the message is different for each node.

The next stage is that the sensor nodes will receive the message from their own adjacent nodes. For instance, if a sensor node has 3 adjacent nodes, this sensor node will receive 3 different messages. Each message will then be encrypted and forwarded to the base station. The sensor node will do 3 times as there are 3 different adjacent nodes.

In the final stage, the base station will decrypt the messages and display the output to a log file. Similar to the sensor node sending forward the message 3 times, the base station will receive 3 different messages for decryption.

IMAGE 1



III. IPC ALGORITHM DESIGN SCHEME

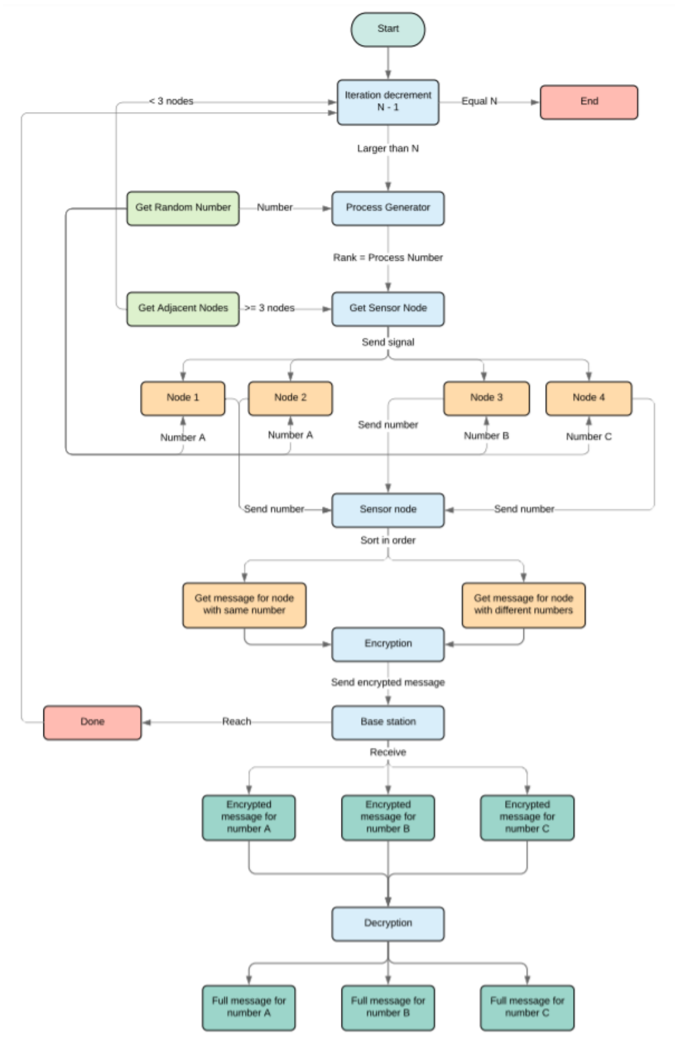
In the IPC algorithm using C and MPI, the number of triggered events (3 or more adjacent nodes near the sensor node) occurs is tied to the number of iterations.

For each iteration, a random number from 1 to 20 (based on the width and height of the table) is generated by the process generator. This number indicates the sensor node that is about to be examined for the event detection. In order to retrieve this node, it must be used with MPI_Comm_rank to retrieve the rank equal to the node value generated by the process generator.

The sensor then sends the signals to its adjacent nodes. Receiving the signal, the nodes with their corresponding random numbers send back these numbers to the sensor node. It sorts the random numbers into the ascending order with repeating number will be just one to eliminate duplicates. Each number forms part of the message to be further encrypted.

The message is encrypted with Caesar cipher and sent to the base station to be decrypted. Once the original message is recovered, it is printed out in a log file. The whole process continues depending on the number of iterations set at the start of the program.

IMAGE 2

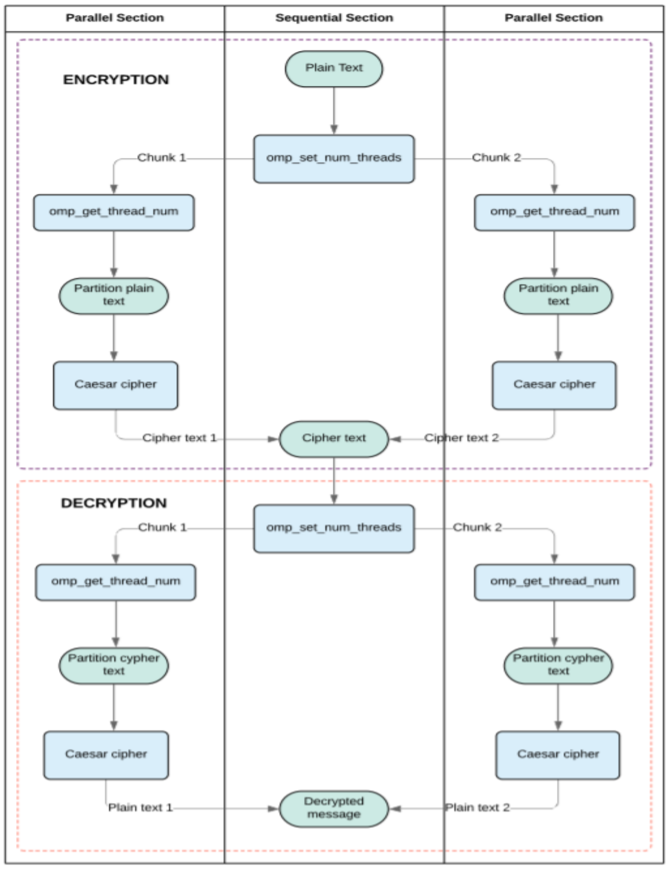


IV. ENCRYPTION ALGORITHM

The encryption and decryption design is done with 2 parallel threads and the algorithm-in-use is Caesar cipher. When the message is partitioned into 2 sections for the 2 threads. According to image 2, there are 2 chunks which represent the length of the partitioned message. This chunk determines the starting and ending of the text is about to be encrypted. Once the cipher text is produced by Caesar cipher, they are combined with the remaining cipher text from the second chunk. This whole process is similar to decryption which is the opposite of encryption.

Encryption in Caesar cipher works with each character being replaced by another character. The replacement is determined by the number of jumps. For instance, if the jump is 3, the current character is ‘a’, then the character replaces this character is ‘d’ because ‘a’ is 3-step behind ‘d’. This is done for the rest of the text. Decryption is about reversing the characters back to its original using the same jump number [2].

IMAGE 3

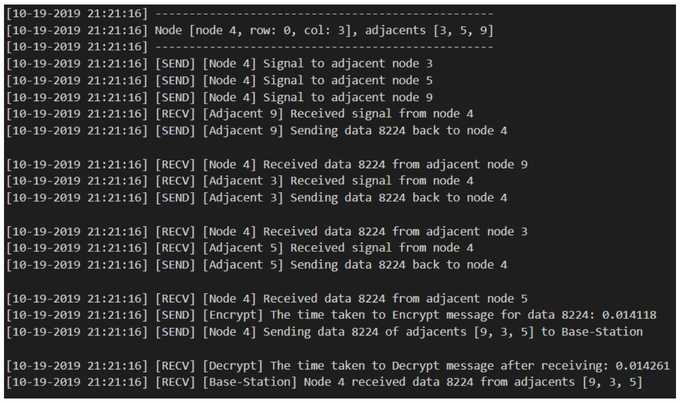


V. RESULTS AND DISCUSSION

Compared to the hypothesis and prediction in Table 1 above, the time taken is generally not much faster when applying OpenMP to encrypt the message. However, it can be clearly seen in Table 3 that the time taken to decrypt the message is much faster. Each message takes around 0.0001 seconds to decrypt with OpenMP while without OpenMP it takes 0.005 seconds.

There have been some issues such as the adjacent nodes receiving the same random numbers happening in all some or all iterations, . When working in pure C codes, the random number function can generate numbers randomly. This cannot be done the same in MPI. The problem arises can be due to all nodes are running parallel thus one random number are assigned to all of those nodes. A minor adjustment is required to make sure that all adjacent nodes receive a number based on their ranks. Although this does not affect the communication time, but it affected the testing to confirm the correctness of the whole program to send the data based on the random number.

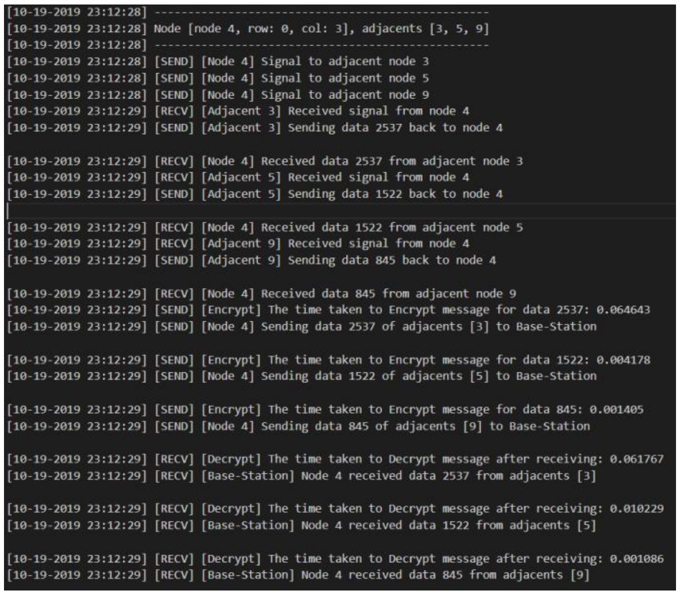
IMAGE 4



```
[10-19-2019 21:21:16] -------------------------------------------------
[10-19-2019 21:21:16] Node [node 4, row: 0, col: 3], adjacents [3, 5, 9]
[10-19-2019 21:21:16] -------------------------------------------------
[10-19-2019 21:21:16] [SEND] [Node 4] Signal to adjacent node 3
[10-19-2019 21:21:16] [SEND] [Node 4] Signal to adjacent node 5
[10-19-2019 21:21:16] [SEND] [Node 4] Signal to adjacent node 9
[10-19-2019 21:21:16] [RECV] [Adjacent 9] Received signal from node 4
[10-19-2019 21:21:16] [SEND] [Adjacent 9] Sending data 8224 back to node 4

[10-19-2019 21:21:16] [RECV] [Node 4] Received data 8224 from adjacent node 9
[10-19-2019 21:21:16] [RECV] [Adjacent 3] Received signal from node 4
[10-19-2019 21:21:16] [SEND] [Adjacent 3] Sending data 8224 back to node 4

[10-19-2019 21:21:16] [RECV] [Node 4] Received data 8224 from adjacent node 3
[10-19-2019 21:21:16] [RECV] [Adjacent 5] Received signal from node 4
[10-19-2019 21:21:16] [SEND] [Adjacent 5] Sending data 8224 back to node 4

[10-19-2019 21:21:16] [RECV] [Node 4] Received data 8224 from adjacent node 5
[10-19-2019 21:21:16] [SEND] [Encrypt] The time taken to Encrypt message for data 8224: 0.014118
[10-19-2019 21:21:16] [SEND] [Node 4] Sending data 8224 of adjacents [9, 3, 5] to Base-Station

[10-19-2019 21:21:16] [RECV] [Decrypt] The time taken to Decrypt message after receiving: 0.014261
[10-19-2019 21:21:16] [RECV] [Base-Station] Node 4 received data 8224 from adjacents [9, 3, 5]
```

The above image illustrates the communication time between the sensor node and the base station. In this case, the sensor node only sends the data to the base station once because of all adjacent nodes receiving the same random number.

IMAGE 5



```
[10-19-2019 23:12:28] -------------------------------------------------
[10-19-2019 23:12:28] Node [node 4, row: 0, col: 3], adjacents [3, 5, 9]
[10-19-2019 23:12:28] -------------------------------------------------
[10-19-2019 23:12:28] [SEND] [Node 4] Signal to adjacent node 3
[10-19-2019 23:12:28] [SEND] [Node 4] Signal to adjacent node 5
[10-19-2019 23:12:28] [SEND] [Node 4] Signal to adjacent node 9
[10-19-2019 23:12:29] [RECV] [Adjacent 3] Received signal from node 4
[10-19-2019 23:12:29] [SEND] [Adjacent 3] Sending data 2537 back to node 4

[10-19-2019 23:12:29] [RECV] [Node 4] Received data 2537 from adjacent node 3
[10-19-2019 23:12:29] [RECV] [Adjacent 5] Received signal from node 4
[10-19-2019 23:12:29] [SEND] [Adjacent 5] Sending data 1522 back to node 4

[10-19-2019 23:12:29] [RECV] [Node 4] Received data 1522 from adjacent node 5
[10-19-2019 23:12:29] [RECV] [Adjacent 9] Received signal from node 4
[10-19-2019 23:12:29] [SEND] [Adjacent 9] Sending data 845 back to node 4

[10-19-2019 23:12:29] [RECV] [Node 4] Received data 845 from adjacent node 9
[10-19-2019 23:12:29] [SEND] [Encrypt] The time taken to Encrypt message for data 2537: 0.064643
[10-19-2019 23:12:29] [SEND] [Node 4] Sending data 2537 of adjacents [3] to Base-Station

[10-19-2019 23:12:29] [SEND] [Encrypt] The time taken to Encrypt message for data 1522: 0.004178
[10-19-2019 23:12:29] [SEND] [Node 4] Sending data 1522 of adjacents [5] to Base-Station

[10-19-2019 23:12:29] [SEND] [Encrypt] The time taken to Encrypt message for data 845: 0.001405
[10-19-2019 23:12:29] [SEND] [Node 4] Sending data 845 of adjacents [9] to Base-Station

[10-19-2019 23:12:29] [RECV] [Decrypt] The time taken to Decrypt message after receiving: 0.061767
[10-19-2019 23:12:29] [RECV] [Base-Station] Node 4 received data 2537 from adjacents [3]

[10-19-2019 23:12:29] [RECV] [Decrypt] The time taken to Decrypt message after receiving: 0.010229
[10-19-2019 23:12:29] [RECV] [Base-Station] Node 4 received data 1522 from adjacents [5]

[10-19-2019 23:12:29] [RECV] [Decrypt] The time taken to Decrypt message after receiving: 0.001086
[10-19-2019 23:12:29] [RECV] [Base-Station] Node 4 received data 845 from adjacents [9]
```

In Image 5, due to having different random numbers, each time the sensor node sends the data to the base station, the communication time will vary for that many nodes.

As proposed by the hypothesis for the total number of events, the above images confirm that the number of events is dependent on the number of adjacent nodes. By applying the formula for messages, we can also find the total number of messages in each iteration.

TABLE 3

Attempted run	Iteration	Encryption time (in second)	Decryption time (in second)	Number of messages	Number of events	Event detail
1	1	0.014118	0.014261	7	3	Node [node 4, row: 0, col: 3], adjacents [3, 5, 9]
	2	0.014257	0.000017	7	3	Node [node 6, row: 1, col: 0], adjacents [1, 7, 11]
	3	0.011321	0.000014	7	3	Node [node 11, row: 2, col: 0], adjacents [6, 12, 16]
2	1	0.010373	0.017203	7	3	Node [node 6, row: 1, col: 0], adjacents [1, 7, 11]
	2	0.014588	0.000036	9	4	Node [node 7, row: 1, col: 1], adjacents [2, 6, 8, 12]
	3	0.004909	0.000175	7	3	Node [node 15, row: 2, col: 4], adjacents [10, 14, 20]
3	1	0.0111	0.016021	9	4	Node [node 9, row: 1, col: 3], adjacents [4, 8, 10, 14]
	2	0.014573	0.000016	9	4	Node [node 13, row: 2, col: 2], adjacents [8, 12, 14, 18]
	3	0.000671	0.000099	7	3	Node [node 4, row: 0, col: 3], adjacents [3, 5, 9]

VI. CONCLUSION

In this report, the experiment has shown how the communication among the sensor nodes and the base station works. Such network has been applicable to many fields including environment, health care, security, etc. In health care, for example, by receiving the correct data and information, the emergency service can be updated with urgent cases and provides the necessary care to its patients [3]. In addition, the encryption/decryption will protect the data when it is delivered to the outside network.

References

[1] geeksforgeeks.org, 'Inter Process Communication (IPC)', [Online]. Available: https://www.geeksforgeeks.org/inter-process-communication-ipc/#targetText=Inter%20process%20communication%20(IPC)%20is,Shared%20Memory. [Accessed: 8 – Oct – 2019].

[2] trytoprogram.com, 'C program to encrypt and decrypt the string', [Online]. Available: http://www.trytoprogram.com/c-examples/c-program-to-encrypt-and-decrypt-string/. [Accessed: 12 – Oct – 2019].

[3] A C Djedouboum, A A A Ari, A M Gueroui, A Mohamadou, Z Aliouat, “Big Data Collection in Large-Scale Wireless Sensor Networks”, Sep. 2018, pp. 12