A simulation of fully distributed wireless sensor network

Initial report prepared by:

Cheng Zeng

Abstract— **Wireless sensor network (WSN) is comprised of senor nodes that can collect data and communicate with other sensors and a base station. This report presents an Inter Process Communication architecture MPI Cartesian Topology to implement a simulated distributed wireless sensor network. This WSN is capable of detecting whether a sensor received more than three same numbers from its adjacent nodes. All passing messages through the network are encrypted using XOR algorithm. OpenMP is implemented in XOR algorithm, but it does not improve the performance of encryption.**

Keywords—Wireless sensor network, Inter process communication, MPI, OpenMP, encryption, decryption

I. INTRODUCTION

The objective of the assignment is to simulate a wireless sensor network, in which each WSN node can send a random number to its adjacent nodes and receive a random number from every adjacent node simultaneously. If at least three provisioned numbers are same, an event occurs and a related event report will be sent to the base station by the respective WSN node. The base station is one component of WSN which can communicate with all nodes and write all information to a log file.

Inter process communication (IPC) is a mechanism which allows processes to communicate each other and synchronize their action. There are two ways for processes to communicate with each other: (1) Shared Memory (2) Message passing. IPC is applied to simulate the WSN. All WSN nodes and the base station are represented by a process, the communications between an adjacent node and the base station or among adjacent nodes are established via message passing. For the security reason, all message should be encrypted before sending and decrypted after receiving. In addition, the performance of encryption or decryption using Open Multi-processing (OpenMP) will be tested. The implementation of MSN will be achieved using the Message Passing Interface (MPI) library in C language is used.

II. IPC IMPLEMENTATION

A. IPC Grid Architecture

In the WSN, there are 20 nodes and a base station, they all play a role of process in the IPC architecture. These 20 nodes are arranged in a 4 * 5 grid (see Figure 1), where the base station exists independently. Each node has its own number and also knows its all adjacent nodes' numbers. For each node, its adjacent nodes only include nodes that are in top-bottom and left-right directions. For example, node 0's adjacent nodes are 1 and 5; node 9's adjacent nodes are 4, 8 and 14; node 12's adjacent nodes are 7, 11, 13 and 17 (see Figure 2). Each node can only exchange data with its adjacent node and the base station.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

20 (base station)

Fig. 1 Inter Process Communication Grid and Base Station

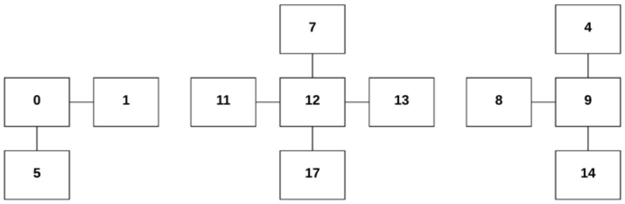
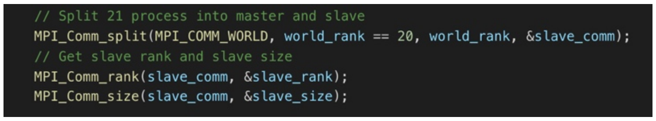


Fig. 2 Adjacent Nodes Examples

At first, a master/slave system is built in order to simulate the IPC architecture. This can be fulfilled by splitting the processes in MPI_COMM_WORLD (i.e. the communicator among all processes) into two parts (shown in Figure 3): (1) the master (i.e. base station) who can decrypt received data and log them (2) the slaves (i.e. nodes) who will send and receive random numbers and contact the master. Besides, the slaves are also able to perform other computation, such as encrypting and decrypting data, detecting event.



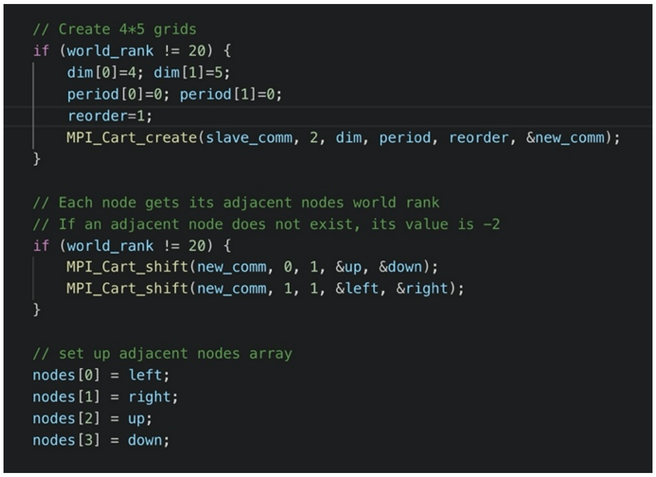
```
// Split 21 process into master and slave
MPI_Comm_split(MPI_COMM_WORLD, world_rank == 20, world_rank, &slave_comm);
// Get slave rank and slave size
MPI_Comm_rank(slave_comm, &slave_rank);
MPI_Comm_size(slave_comm, &slave_size);
```

Fig. 3 C code demonstrating the split of processes

After split, all nodes in the slave set are able to communicate with other nodes via slave_comm which is the communicator among all slaves, where a node can communicate with the base station via MPI_COMM_WORLD (MPI topic: Topologies, n.d.).

Secondly, the 20 slaves process can be arranged in a 4 * 5 grid using Cartesian grid topology. The salve communicator describes the group of slave nodes, but the structure is not such that every node knows its adjacent nodes. A cartesian grid in 2 dimensions is a structure of cells that has neighbors in each dimension. If a Cartesian grid has size of M * N, the cell (m, n) has at most four neighbors (m+1, n), (m-1, n), (m, n+1) and (m, n-1); the exception is the edge points. To implement the

Cartesian grid in the assignment, the Cartesian topology interface is used. MPI_Cart_create creates a new communicator by giving the number of processes in each dimension and whether the processes are periodic in that dimension. MPI_Cart_shift can find the shifted source and destination given the shift direction. Apply these two MPI functions shown in Figure 4, a Cartesian grid of size 4 * 5 is created, and each node has the knowledge of its adjacent nodes. A node will communicate with its adjacent via a new communicator new_comm.



```
// Create 4*5 grids
if (world_rank != 20) {
    dim[0]=4; dim[1]=5;
    period[0]=0; period[1]=0;
    reorder=1;
    MPI_Cart_create(slave_comm, 2, dim, period, reorder, &new_comm);
}

// Each node gets its adjacent nodes world rank
// If an adjacent node does not exist, its value is -2
if (world_rank != 20) {
    MPI_Cart_shift(new_comm, 0, 1, &up, &down);
    MPI_Cart_shift(new_comm, 1, 1, &left, &right);
}

// set up adjacent nodes array
nodes[0] = left;
nodes[1] = right;
nodes[2] = up;
nodes[3] = down;
```

Fig. 4 C code demonstrating the implementation of Cartesian Grid

B. *Event Detection and Encryption/Decryption Algorithm*

In order to simulate the WSN of event detection for a period of time, the message passing process is designed in a for loop, and the number of iterations can be changed. In each iteration, a random number is generated and encrypted by each node in the grid. Every node then will send its encrypted number to its all adjacent nodes. After the node receives all numbers from its adjacent nodes, it starts to decrypt these numbers. Encryption and decryption time are both recorded. An event occurs for a node only if at least three of adjacent nodes send the same numbers. If an occurrence of an event is counted, the node records the date and time, received numbers and their sender. Since the message passing can be expensive in wireless network, it would be more energy efficient to decrease the number of messages passed to base station. Instead of sending messages to the base station every time when an event is detected, a packed data will be sent after iteration ends. Every node will have a pack of data which includes information of the node, the time of encrypting and decrypting messages, the number of messages received from and passed its neighbor nodes, the number of events occurred during the simulation, and the details of each event occurrence. Before the packed data is sent, it will be encrypted by its owner. At the end of simulation, the base station receives 20 packed data, then decrypt them and write them in a file.

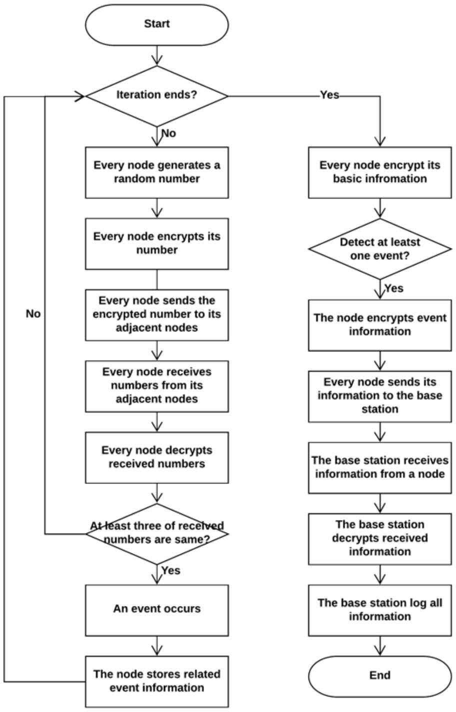

Start
Iteration ends?
Yes
No
Every node generates a random number
Every node encrypts its number
Every node sends the encrypted number to its adjacent nodes
Every node receives numbers from its adjacent nodes
Every node decrypts received numbers
At least three of received numbers are same?
No
Yes
An event occurs
The node stores related event information
Every node encrypt its basic infromation
Detect at leatst one event?
Yes
The node encrypts event information
Every node sends its information to the base station
The base station receives information from a node
The base station decrypts received information
The base station log all information
End

Fig. 5 Flowchart of Event Detection in a WSN

In a WSN, each sensor is capable of collecting information, communicating with other sensors and forwarding data to a base station. The security is always a potential issue in the field of data transmission, especially via the network. In this assignment, a simple encryption algorithm XOR Cipher is applied (XOR cipher, 2019). The principles of XOR encryption algorithm are:

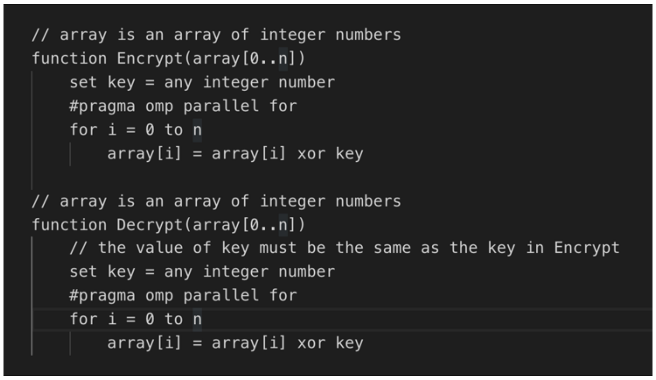
$$\begin{array}{l} 1 \text{ xor } 0 = 1 \\ 0 \text{ xor } 1 = 1 \\ 1 \text{ xor } 1 = 0 \\ 0 \text{ xor } 0 = 0 \end{array}$$

Every decimal number can be represented as a binary number, so xor operation can be performed on two integer numbers which can be easily converted into its binary number. For example, two integer numbers 17 and 37 can be converted to binary numbers 010001 and 100101, perform xor operation on these two binary numbers to get another binary number 110100 which is 52 in decimal:

$$\begin{array}{rr} & 0\,1\,0\,0\,0\,1 \\ \text{xor} & \underline{1\,0\,0\,1\,0\,1} \\ & 1\,1\,0\,1\,0\,0 \end{array}$$

To implement XOR encryption in the assignment, an encryption key is first defined and then perform xor operation on the target number with the key. And conversely, to decrypt the encrypted number, the xor operation is performed again with the encryption key. In this simulation of WSN, all passing messages are converted into integers in order to be encrypted using this XOR encryption.

OpenMP is an application programming interface (API) which implements multithreading to support multiprocessing programming (OpenMP, 2019). The principle of multithreading is that a master thread forks off a specified number of threads and then allocate different sub-tasks to them. Parallelism can be achieved because each thread can perform sub-tasks concurrently. At the end of the simulation of WSN, each node will encrypt and send a big packed data to the base station. If a large number of events occurred, the base station may end up decrypting a huge amount of information that are sent from 20 nodes. Speeding up the process of encryption and decryption may be able to be achieved by implementing OpenMP in the encryption or decryption algorithm. This is because the whole data can be divided into chunks that are allocated to different threads. Then threads can perform encryption or decryption on their own chunk of data. The pseudo code is shown in Figure 6. To apply OpenMP in the encryption or decryption algorithm, a line *#pragma omp parallel for* needs be.



```
// array is an array of integer numbers
function Encrypt(array[0..n])
    set key = any integer number
    #pragma omp parallel for
    for i = 0 to n
        array[i] = array[i] xor key

// array is an array of integer numbers
function Decrypt(array[0..n])
    // the value of key must be the same as the key in Encrypt
    set key = any integer number
    #pragma omp parallel for
    for i = 0 to n
        array[i] = array[i] xor key
```

Fig. 6 Pseudo Code of Encryption/Decryption algorithm

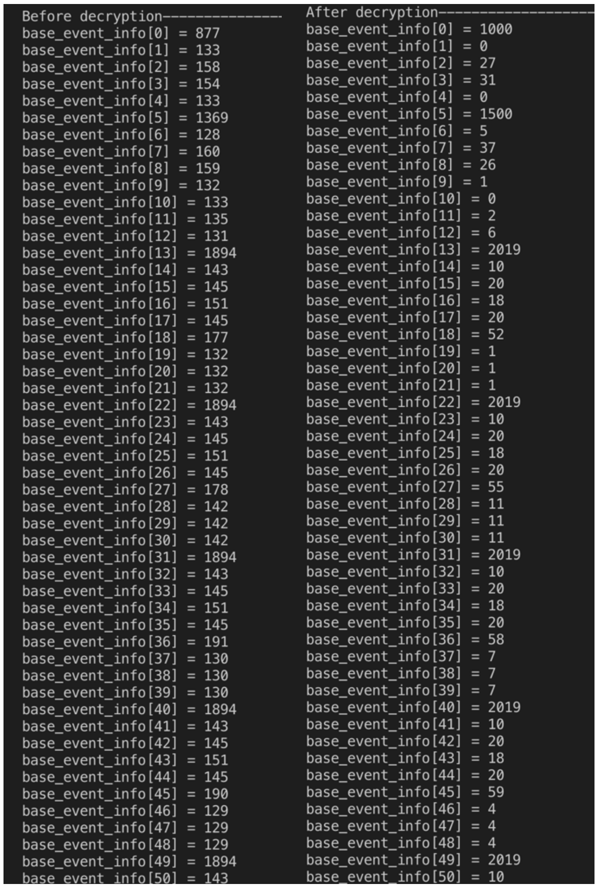
III. RESULTS AND DISSCUSSIONS

The simulation using serial encryption algorithm and OpenMP encryption was performed five times respectively. All information is logged in to result file. Some valuable data is recorded in Table A and Table B in the appendices.

The implementation of encryption/decryption using serial and OpenMP both succeed in encrypt data, a small part of messages which is received by the base station is shown in Figure 7, the left half is messages before decryption and the right half is messages after decryption.

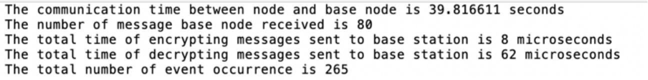
In Appendix Table A and Table B, no obvious speed up of encryption or decryption is observed according to average time. This issue could happen when the simulation runs on a single computer, which cannot afford enough logical processors to execute 21 MPI processes in parallel. However, the total time of decrypting messages passed among nodes in the third and five runs are indeed shorter, they are both about one fifth of the average time. This might show that OpenMP can speed up the encryption/decryption.

The total communication time between nodes and the base station and that between a node and its adjacent nodes are recorded in the log file, Figure 8 and Figure 9 show base station communication time and node 7 communication time respectively. The result shows the time of the base station receiving 80 messages from nodes is almost same as the time of node 7 passing 4000 messages between its adjacent nodes. This is because all nodes communicate with the base station via the same communicator. If different nodes try to send data to the base station at the same time, some of communications must be suspended which caused more communication time.



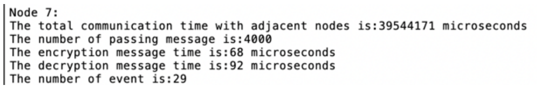
```
Before decryption---------------    After decryption-------------------
base_event_info[0] = 877            base_event_info[0] = 1000
base_event_info[1] = 133            base_event_info[1] = 0
base_event_info[2] = 158            base_event_info[2] = 27
base_event_info[3] = 154            base_event_info[3] = 31
base_event_info[4] = 133            base_event_info[4] = 0
base_event_info[5] = 1369           base_event_info[5] = 1500
base_event_info[6] = 128            base_event_info[6] = 5
base_event_info[7] = 160            base_event_info[7] = 37
base_event_info[8] = 159            base_event_info[8] = 26
base_event_info[9] = 132            base_event_info[9] = 1
base_event_info[10] = 133           base_event_info[10] = 0
base_event_info[11] = 135           base_event_info[11] = 2
base_event_info[12] = 131           base_event_info[12] = 6
base_event_info[13] = 1894          base_event_info[13] = 2019
base_event_info[14] = 143           base_event_info[14] = 10
base_event_info[15] = 145           base_event_info[15] = 20
base_event_info[16] = 151           base_event_info[16] = 18
base_event_info[17] = 145           base_event_info[17] = 20
base_event_info[18] = 177           base_event_info[18] = 52
base_event_info[19] = 132           base_event_info[19] = 1
base_event_info[20] = 132           base_event_info[20] = 1
base_event_info[21] = 132           base_event_info[21] = 1
base_event_info[22] = 1894          base_event_info[22] = 2019
base_event_info[23] = 143           base_event_info[23] = 10
base_event_info[24] = 145           base_event_info[24] = 20
base_event_info[25] = 151           base_event_info[25] = 18
base_event_info[26] = 145           base_event_info[26] = 20
base_event_info[27] = 178           base_event_info[27] = 55
base_event_info[28] = 142           base_event_info[28] = 11
base_event_info[29] = 142           base_event_info[29] = 11
base_event_info[30] = 142           base_event_info[30] = 11
base_event_info[31] = 1894          base_event_info[31] = 2019
base_event_info[32] = 143           base_event_info[32] = 10
base_event_info[33] = 145           base_event_info[33] = 20
base_event_info[34] = 151           base_event_info[34] = 18
base_event_info[35] = 145           base_event_info[35] = 20
base_event_info[36] = 191           base_event_info[36] = 58
base_event_info[37] = 130           base_event_info[37] = 7
base_event_info[38] = 130           base_event_info[38] = 7
base_event_info[39] = 130           base_event_info[39] = 7
base_event_info[40] = 1894          base_event_info[40] = 2019
base_event_info[41] = 143           base_event_info[41] = 10
base_event_info[42] = 145           base_event_info[42] = 20
base_event_info[43] = 151           base_event_info[43] = 18
base_event_info[44] = 145           base_event_info[44] = 20
base_event_info[45] = 190           base_event_info[45] = 59
base_event_info[46] = 129           base_event_info[46] = 4
base_event_info[47] = 129           base_event_info[47] = 4
base_event_info[48] = 129           base_event_info[48] = 4
base_event_info[49] = 1894          base_event_info[49] = 2019
base_event_info[50] = 143           base_event_info[50] = 10
```

Fig. 7 A Small part of Messages Logs Before and After Decryption



```
The communication time between node and base node is 39.816611 seconds
The number of message base node received is 80
The total time of encrypting messages sent to base station is 8 microseconds
The total time of decrypting messages sent to base station is 62 microseconds
The total number of event occurrence is 265
```

Fig. 8 Communication Time between nodes and base station



```
Node 7:
The total communication time with adjacent nodes is:39544171 microseconds
The number of passing message is:4000
The encryption message time is:68 microseconds
The decryption message time is:92 microseconds
The number of event is:29
```

Fig. 9 Communication Time between node 7 and its adjacent nodes

IV. CONCLUSIONS

The report presents a simulation of a WSN which can detects a specified event. The inter process communication of message passing is applied and a cartesian grid of 4 * 5 is created to simulate the target WSN. For security issue, XOR cipher is chosen to encrypt and decrypt all passing messages. OpenMP is used to analyze the speeding up of encryption and decryption.

In conclusion, no obvious speeding up is observed according to the average time of encryption and decryption. However, a couple of execution results show that decryption time is much shorter than the average time. To further verify the performance of speeding up encryption or decryption algorithm which implements OpenMP, this program can be tested on MONARCH.

References

MPI topic: Topologies. (n.d.). Retrieved from http://pages.tacc.utexas.edu/~eijkhout/pcse/html/mpi-topo.html

OpenMP. (2019). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/OpenMP

Wireless sensor network. (2019). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Wireless_sensor_network

XOR cipher. (2019). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/XOR_cipher

APPENDIX

TABLE A (SERIAL IMPLEMENTATION)
** All times are expressed in microseconds

Value of Iteration	1000				
Range of random number	1-12				
	Serial encryption/encryption				
	Number of event occurrences	The total time of encrypting messages sent to the base station from 20 nodes	The time of decrypting messages sent to the base station	The total time of encrypting messages passed among nodes	The total time of decrypting messages passed among nodes
Run # 1	230	5	65	966	110831909
Run # 2	219	6	67	1276	147712702
Run # 3	215	4	64	1229	278227218
Run # 4	203	6	65	1455	384777253
Run # 5	242	11	69	1480	173997082
Average	221.8	6.4	66	1281.2	219109233

TABLE B (OPENMP IMPLEMENTATION)
** All times are expressed in microseconds

Value of Iteration	1000				
Range of random number	1-12				
	OpenMP encryption/decryption				
	Number of event occurrences	The total time of encrypting messages sent to the base station from 20 nodes	The time of decrypting messages sent to the base station	The total time of encrypting messages passed among nodes	The total time of decrypting messages passed among nodes
Run # 1	246	8	65	1220	398010977
Run # 2	231	8	65	1304	287820003
Run # 3	254	6	63	1264	42453531
Run # 4	250	8	68	1147	364276067
Run # 5	259	7	64	1192	41445686
Average	248	7.4	65	1225.4	226801253