Optimising Inter Process Communication in a Wireless Sensor Network with a Grid with Apex Node Topology

Tarik Pecaninovic
Faculty of Information Technology
Monash University, Clayton

ABSTRACT Simulation and analysis of a Wireless Sensor Network (WSN) with a given topology and further constraints on that given topology are explored in this paper with the express purpose of optimising the efficiency of the Inter Process Communication (IPC). The network topology used is that of a grid with a single apex node; furthermore the given constraint on the topology is that nodes within the grid may only communicate with adjacent nodes and the apex node (while the apex node may communicate with every node). Message Passing Interface (MPI) is used as the communication method between nodes and OpenMP is used to simulate an extra node.

I. INTRODUCTION

Wireless Sensor Networks (WSN) are widespread in their use in many applications across the globe [1].
In this paper we explore a WSN where the network has the following topology: every vertex in a mxn node is connected to a single node outside of this grid (this node is termed an apex node). Furthermore, to this topology the following constraint is added: each node in the grid may only communicate with its immediate neighbours and the apex node.

The problem which was simulated using this network was the following: A grid of nodes randomly generates a positive integer (which represents a given temperature in that region) during a given time window. When a node generates a positive integer which is greater than some given threshold value, the node checks to see if there are at least two neighbouring nodes which also generated temperatures greater than the given threshold (within the same time window). If there are, then the original node signals a base station node with this information. The base station then compares these values with the values generated by a node which represents a satellite station (which generates values similarly to the grid nodes). If there is no discrepancy between these values, the base station logs this data (termed an 'event') into a file. Otherwise, it logs the event as a false alert.

The simulation was implemented through the use of the Inter Process Communication (IPC) Message Passing Interface (MPI). Each node in the network is simulated via an MPI process, and moreover the apex process utilizes a POSIX thread (via OpenMP) to simulate a satellite sensor (described below).

We attempted to explore whether there was a way to optimise the efficiency of the communication between the nodes in this experimental setup; hypothesising that there was. We claimed that--through the use of MPI features such as a built-in function which defines a grid network of processes--we would be able to minimize the number of required messages sent in total, and hence improve runtime. The specifics of the design are described in the following section.

II. Design Scheme for Sensor Network

In the following we describe the general design of the network along with the design of the simulation (how communication is handled, how 'events' are detected, how 'events' are logged). For the sake of example and demonstration, in the following we shall consider the specific case of m=3 and n=2.

A. Implementation of the Network and Communication in MPI

Grid networks appear in many branches of applications in many fields [2]. As a result there are many implementations of such networks in readily available implementations of existing specifications. MPI has its own implementation in the form of the function 'MPI_Cart_create'. This function takes in as input an array, where the length is equal to the dimension of the array, where the elements are the corresponding sizes of the dimensions. Using our example then, the input shall be [3,2].

This function defines a new communicator for nodes within the grid which provides an efficient form of

communication between these nodes. We shall make extensive use of this throughout, as it will allow us to 'group' together adjacent nodes. The utilization of these 'groups' of nodes within the grid will substantially reduce the number of messages required to be sent--hence improving runtime.

The apex node shall be implemented independently as its own process as it only needs to communicate with individual nodes in the grid when needed--meaning a naive approach will not significantly impact performance. Standard functions, such as MPI_Sendrect, shall be utilized for communication in this part of the network.

Finally, the last 'virtual' node, which represents the satellite sensor, shall be implemented using a POSIX thread in the apex node process. Again, as communication is only between two nodes for this portion, a naive implementation will not significantly impact performance. The POSIX thread will be implemented via the use of OpemMP.

The overall topology/architecture of the network is depicted in Fig. 1.

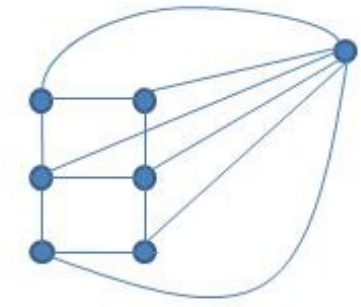

FIGURE 1. **A depiction of the network architecture for the m=3, n=2 case. The grid nodes can be seen on the left--these all belong to the same communicator defined by the MPI function. The apex node can be seen on the upper left hand corner. Note that an edge here represents a two way communication path.**

B. Event Detection

We discuss the implementation of 'Event Detection'. To determine if an event has occurred, a node needs to generate a value which is greater than a predefined threshold, and then compare this value with the value of other nodes in the network, which were generated in the same time window.The main point which needs to be addressed is that of nodes comparing values. This is because each node generates values independently of any other node.

The design we went for is the following: When a node generates a value it includes a corresponding timestamp. Then, when it gets a request from another node to send its value to compare (note the sending node will send the timestamp of its value), it finds the value with the timestamp which is closest to the received one. Note that as each value is generated after a given time (determined by the time window), there will only be one timestamp to return to the node.

B. Overall Algorithms

Algorithm 1 Describes the algorithm each grid node follows during run-time	
1	While (no termination signal received from base):
2	During given time window:
3	Generate temperature
4a	If temperature is greater than predefined threshold: Compare with adjacent nodes temperatures. Else: Repeat from Step 1
4b	If at least two adjacent nodes temperature values agree: Send 'event' to base station node. Else: Repeat from Step 1

Algorithm 2 Describes the algorithm the base station node follows during run-time	
1	For a fixed, large, number of iterations (after which, send a termination signal to grid nodes):
2	For a fixed time window:
3	Attempt to receive messages from grid nodes, while in parallel (using a POSIX thread) generate a temperature for the satellite sensor.
4	If a message is received from a grid node: Compare its value with the generated satellite value. Else: Repeat from Step 1.
5	If grid node value agrees with satellite value: Log event to file. Else: Log event to file as 'False Alert'.

The algorithms described in '**Algorithm 1**' and '**Algorithm 2**' provide high level overviews of the algorithms executed by the grid and base station nodes, moreover implementing the design features described in the previous subsections.

III. Results and Discussion

Unfortunately, due to personal circumstances we were not able to complete the implementation of the design in C. Hence, we are only able to discuss the theoretical improvements the presented design provides.

The major advantage our design provides over a native implementation is that of improved efficiency. A major bottleneck a naive implementation would encounter is that of communication time within the nodes in the grid--specifically during event detection. This is due to the large number of messages being sent in a naive implementation. In our design we minimize this number through the use of MPI. As nodes in the grid are grouped together with their adjacent neighbours, we are able to more efficiently send messages to the nodes within these groups.

Another advantage our design provides is that of ease of use. As we generate the entire grid cluster using MPI, this grid cluster can be seen as one object, and hence is abstracted. This abstraction leads to a simplified setup

IV. Conclusion

To recapitulate the main points of this report: We wished to see whether we could implement a WSN (given a specific problem) using the tools available in MPI and OpenMP, and moreover focus on the efficiency of the IPC by minimizing the number of messages sent during runtime. This would in essence be a proof of concept for the existence of such a network, and moreover in the future could be used to draw comparisons between implementations of other such WSNs.

In designing the network and the general algorithm for the problem (which included the communication between them), the main points considered include: To minimize the number of messages being sent between adjacent nodes, the in-built MPI function which creates a ‘grid network’ of processes was used, as this allowed us to group nodes together by their ‘adjacency’; this then allowing us to minimize the number of messages being sent between these nodes. We implemented a naive method for both communication between the apex node and the grid nodes and the satellite node, as we hypothesized this would have little impact on the overall performance.

Although we were not able to finish the implementation of the design due to outside circumstances, we did discuss results found in similar papers outside of this one, and moreover discussed the theoretical advantages of our design. We found the main advantage to be: The abstraction of the nodes within the grid leads to much greater ease of use for the end user. The use of both a network of nodes and threads leads to greater efficiency when considering applications.

In conclusion, we found that in theory our design offers a few advantages--including performance and simplicity--over a naive implementation.

REFERENCES

[1] Mazzer, Y. and Tourancheau, B., 2008, September. MPI in wireless sensor networks. In *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting* (pp. 334-339). Springer, Berlin, Heidelberg.

[2] Danner, A., Breslow, A., Baskin, J. and Wilikofsky, D., 2012, November. Hybrid MPI/GPU interpolation for grid DEM construction. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems* (pp. 299-308).