Parallel Routing in Wireless Sensor Network

Prof.Dr.Kais Al Sabbagh
Computer Engineering Department
Engineering College, Baghdad University
Kais_1946@yahoo.com

Dr. Zainab Tawfeeq Baqer
Electrical Engineering Department
Engineering College, Baghdad University
zainab_alisa@yahoo.com

ABSTRACT

The limitations of wireless sensor nodes are power, computational capabilities, and memory. This paper suggests a method to reduce the power consumption by a sensor node .This work is based on the analogy of the routing problem to distribute an electrical field in a physical media with a given density of charges. From this analogy a set of partial differential equations (Poisson's equation) is obtained. A finite difference method is utilized to solve this set numerically. Then a parallel implementation is presented. The parallel implementation is based on domain decomposition, where the original calculation domain is decomposed into several blocks, each of which given to a processing element. All nodes then execute computations in parallel, each node on its associated sub-domain. With this method power consumption by the central node which is responsible to compute routing in the network is reduced.

Keywords- sensor network; electrostatic theory; finite difference method; successive over relaxation; domain decomposition;

التوجيه المتوازي للشبكات التحسس اللاسلكية

الخلاصة

احد أوجه قصور العقد التحسس اللاسلكية هو الطاقة والقدرات الحسابية والذاكرة. وتقترح هذه المقالة طريقة لتقليل استهلاك الطاقة من قبل عقده جهاز التحسس.ويستند هذا العمل الى قياس مشكلة التوجيـه علـى توزيـع الحقـل الكهربائي في وسائل إعلام مادية بكثافة معينة من الشحنات. من هذا القياس تم الحصول علـى مجموعـة مـن المعادلات التفاضلية الجزئية (المعادلة لـ Poisson). واستخدم أسلوب فرق محدودة لحل هذه المجموعة عـددياً. ثم التنفيذالمتوازي قدم. ويستند تنفيذ متوازي على تحلل المجال، حيث يتم حساب المجال الأصلي متحللة إلى كتـل عدة، كل من التي تعطي لعنصر تجهيز. ثم تنفيذ العقد جميع العمليات الحسابية بالتوازي، كل عقده علـى جـزء المجال المرتبطة به. بهذه الطريقة يتم تقليل استهلاك الطاقة بالعقدة المركزية المسؤولة لحساب التوجيه في الشبكة.

الكلمات الرئيسية:الشبكات التحسسية ، نظرية الكهروستاتيكية، طريقة الفرق المحدد، التوالي على الاسـترخاء، تحلل المجال

INTRODUCTION

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices (several hundred to several thousand devices) that use sensors to monitor physical or environmental conditions. These autonomous devices, or nodes, combine with routers and a gateway to create a typical WSN system. The distributed measurement nodes communicate wirelessly to a central gateway, which provides a connection to the wired world where data measurement can be collected, processed, analyzed, and presented. To extend distance and reliability in a wireless sensor network, routers can be used to gain an additional communication link between end nodes and the gateway. The primary areas in which these networks are environmental observations, military monitoring, building monitoring and health care [Vijay 2007].

In many applications the sensors perform measurements of specific metrics such as temperature, pressure, movements or other physical values, and it is desired to collect the data of sensors in a specific station for processing, archiving and other purposes. This station is a data sink, and it has enough processing power, storage space, and capability of communicating with the sensors. For the purpose of communication to the sink, the sensors relay the packets of each other in a multi-hop way.

A novel approach for the routing problem in wireless ad hoc networks was introduced [Kalantari 2004]. This approach is based on the analogy of the routing problem to the distribution of electric field in a physical media with a given density of charges. Then this approach is used in wireless sensor network [Kalantari 2006]. The mathematical model is explained in the following section briefly with some assumptions and consideration. Mathematical details formulation is given by Kalantari [2006] . The work presented in this paper is based on this approach. A Poisson's equations are obtained from this approach. These equations are solved numerically using the finite difference method. This method is parallelized. Because the energy is very important, partitioning the computations between more than one node in the sensor network will achieve better performance. One node needs more time for calculations and distributions the results to the other nodes. So, to reduce this time more nodes are utilized.

MATHMATICAL MODEL

Consider a network with *N* wireless nodes that can communicate with each other through radio links. The

nodes are randomly placed in a region *A* in the plane. Assume there are *M* source-destination pairs, denoted $s_1 \dots\ s_M$. Source-destination pair s_i has a bandwidth demand which we refer to as its *weight* and denote by W_i. Suppose that one or more paths in the plane are chosen for each s_i. Each path starts at the source node and ends at the destination node of s_i. The weight W_i is partitioned into amounts that are assigned to the paths. The weight assigned to a particular path indicates the amount of demand that is desired to follow that path. It should be noted that the chosen paths are not constrained by the location of intermediate nodes. Instead, the paths are 'abstract' paths in the plane that represent desired paths for the transit of packets. For communication to occur, each abstract path must be approximated by an actual path consisting of a piecewise linear multihop path connecting the source and destination through a sequence of intermediate nodes. Given a set of weighted (abstract) paths for each source-destination pair, define a vector field on *A* which refers to as the *load density* vector field and denote by $\vec{D}$. $\vec{D}$ represents the flux density of the weighted paths for the source-destination pairs. Given a

point $(x, y) \in A$, a small area element at (x, y) is

chosen. For each path that intersects *S*, we take the tangent vector to the path and scale it so it has magnitude equal to the weight of the path. Adding up these scaled tangent vectors, dividing by the area of *S*, and letting the area element go to zero gives the value of $\vec{D}$ at (x, y). A problem with this definition is that since there are only finitely many source-destination pairs, and hence only finitely many paths $\vec{D}$ will be 0 except on a set of measure zero.

For optimal $\vec{D}^*$ a set of partial equations can be written

$$\vec{\nabla}.\vec{D}^* = \rho \qquad \vec{\nabla} \times \vec{D}^* = 0 \tag{1}$$

ρ represents the density of sources in the network and $\vec{\nabla}$ is defined as:

$$\vec{\nabla} = \frac{\partial}{\partial x}\hat{i} + \frac{\partial}{\partial y}\hat{j} \tag{2}$$

The above equations are similar to Maxwell's equations in the electrostatic theory. It is proved that the vector field D can be expressed as the gradient of a scalar field. In other wards:

$$\vec{D} = \vec{\nabla} U \tag{3}$$

in which U is a scalar function known as the potential function. Then the set of equations defined by (1) reduces to:

$$\nabla^2 U = \rho \tag{4}$$

in which the operator ∇^2 is defined as:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{5}$$

The partial differential equation defined by (4) is known as the Poisson equation. The potential function gives a rough idea of how much effort by the network is needed to send data from a source to the destination. This effort is proportional to the potential difference of the source and the destination. In addition, the potential function gives insight into the routing. Based on (3), the routing is done in the direction of the gradient of the potential function.

FINITE DIFFERENCE METHOD

The finite differenc method is a numerical analysis technique for obtaining approximate solutions to a wide variety of engineering problems. In many engineering problems, it is difficult to find the analytical solution. So several approximate numerical analysis methods have evolved over the years such as the finite difference method which is a powerful and a versatile numerical technique for handling problems involving complex solution regions. It involves three steps:

(1) Dividing the solution region into a grid of nodes (Fig.1).

(2) Approximating the given differential equation by finite difference equivalent that relates the dependent variable at a point in the solution region to its values at the neighboring points.

(3) Solving the difference equations subject to the prescribed boundary conditions and/or initial conditions.

More information on finite difference method can be found in [Sadiku 2001]. As explained in section II, it is desired to solve the following equation:

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \rho \tag{6}$$

The function U(x,y) in a 2-D domain, x in [0,1] and y in [0,1] is solved. The boundary conditions (U (0, y), U (1, y), U(x, 0), and U(x, 1)) are given. In addition, the function U(x,y) can also be specified on specific points in the interior domain. The source ρ (x,y) is taken to be a simple Gaussian in what follows for simplicity.

The domain is discretized on a 2-D numerical lattice, with $x_i = i * h$, $y_j = j * h$ and $h = \frac{1}{N}$. N is the number of grid points in both directions and h is the mesh size. In this work different size of grids is utilized. The Poisson equation is re-written in finite difference form (five stencil form):

$$\frac{1}{h^2}=(U_{i+1,j}+U_{i-1,j}+U_{i,j+1}+U_{i,j-1}-4*U_{i,j})=\rho_{i,j} \qquad (7)$$

$U_{i,j}$ is the value of U at node (*i,j*) in the mesh for Poisson's equation (**Fig.1**).

This leads to a large set of linear equations to solve for the field values $U_{i,j}$ on the grid within the domain, keeping $U_{i,j}$ fixed at the boundary. This set of equations can be solved by an iterative method, whereby

$$U_{i,j}^{k+1}=\frac{1}{4}(U_{i+1,j}^{k}+U_{i-1,j}^{k}+U_{i,j+1}^{k}+U_{i,j-1}^{k}-h^2\rho_{i,j}) \qquad (8)$$

is iterated until the changes in $U_{i,j}$ become less than some predefined tolerance criteria. An arbitrary guess for $U_{i,j}$ is assumed to start with. We choose $U^0{}_{i,j}$=0, or can be the average of U at the fixed node. The old and new values of $U_{i,j}$ are mixed to accelerate the convergence, leading to the "Successive Over Relaxation" scheme [Azali 2009]. It may define residual $R_{i,j}$ as:

$$R_{i,j}^{k}=(U_{i+1,j}^{k}+U_{i-1,j}^{k}+U_{i,j+1}^{k}+U_{i,j-1}^{k}-4U_{i,j}^{k}-h^2\rho_{i,j}) \qquad (9)$$

$$U_{i,j}^{k+1}=U_{i,j}^{k}+\frac{w}{4}R_{i,j}^{k} \qquad (10)$$

The relaxation factor *w* is selected such that 1< *w* < 2. The choice of a proper value of *w* is problem dependent and is often determined by trial and error. In this paper, *w*=1.5.

The value of the potential function *U* at each node in the mesh is computed as shown in **Fig.2**. Let us assume that the source of packets in the middle of the network ($node_{60,60}$) and it (source) want to broadcast its packets (assuming all the other nodes destinations). The number of nodes in the network=14400 (120x120). In this simulation, the time is recorded for different number of nodes as shown in **Fig.3** with different values of tolerance. As shown, the time is increased when the number of nodes in the network is increased. More accurate results are obtained due to the increases of nodes. Of course increasing the number of nodes gives an accurate results but the time increases. Requiring more time in wireless sensor network make the sensor nodes consume more power. So in this work the original domain is divided into sub-domains in order to distribute the computation between nodes as described in the following section.

PARALLELIZATION STRATEGY

Parallelism in this system is to sub-divide the data structures into sub-domains and assign each sub-domain to one processor. In this case, the same code runs on all processors with its own set of data. By dividing the computational domain into four sub-domains as shown in **Fig.4**, it is possible to spread the workload between four different processors. However, it is important to note that, in order to compute the variables for each nodal point, the variables at its neighboring points are required. Thus, in order to calculate the variables at the points close to the interface between sub-domains, one processor will require information stored in the memory of a neighboring processor. This requires communication at regular intervals, which may slow down the computation. In general, the computation procedure involves three steps (1) partitioning of the solution domain; (2) performing computations on each processor to update its own data set; (3) communicating data between processors. This technique is called domain decomposition. The key for an efficient computation is to maintain the communications between processors to a minimum level, as well as, to divide the workload equally between processors.

In this work, domain decomposition coordinate bisection is used [Andreas 2010]. This method divides the number of points equally between processors, but makes no attempt to obtain a domain division that minimizes the communication between processors, i.e., a division with the smallest number of nodal points in boundaries between sub-domains. Therefore, coordinate bisection may produce sub-domains with long interfaces that will lead to a large amount of communication. This can be partly overcome by recursive application of alternate x, y bisections. The grid is first divided into 2 grids using bisection of the x-length of the calculation domain. Then to each of the resulting domains, y bisection is applied, resulting in four blocks (or sub-domains).The procedure can be continued to obtain eight blocks, sixteen blocks, thirty two blocks, etc. Once a multi-block domain has been established, calculations on each block can begin in parallel if the boundary conditions of these blocks are known. This may be either a physical boundary condition or an internal boundary condition generated as a consequence of the domain decomposition. The physical boundary data of each block, if any, are provided by the user, while the internal boundary data must be received from neighboring blocks, which may reside on different processors. Internal boundary data are hold by buffers on the boundary of each block as shown in **Fig. 4**, which illustrates a calculation sub-domain and the buffer cells used to store the overlap data. Once the buffer data has been received from all sides of a block, the computation of this block can start, using the sequential algorithm. On completion of the solution for the block, the data at its boundaries is sent to the neighboring blocks. Calculation in this block then waits for the buffer update provided by this block's neighbors, after which the next computation cycle can start. The information exchange across sub-domains can be performed using the message passing interface standard, MPI (Message Passing Interface). The use of MPI ensures portability across different computing platforms.

If the communication time is ignored for this method, the time will be decreased with increasing the number of processing elements as shown in **Table 1**. If the suggested model is divided into two sub-domains the time will halved, and if the division into four sub-domains the time will be quarter its origin value and so on as shown in **Fig. 5**. This is true if the communication time is ignored. In fact, with increasing the number of processing elements, the time will be decreased to some threshold value then will be increased due to the overhead appended for communication between the processing elements.

In this work the task is divided into n equal subtasks, each of which can be executed by one processing element. If ts is the time to perform a task by a single processor, tm is the time taken by each processor.

$$tm = ts / n \qquad (11)$$

$$\text{Speed up} = ts / tm = n \qquad (12)$$

This is true when the communication overhead is ignored. With communication overhead:

$$tm = ts /n + tc \qquad (13)$$

where tc is the communication time, So :

$$\text{Speed up} = ts / tm = ts / (ts / n + tc)$$

$$= n / (1+ n^* tc / ts) \qquad (14)$$

For each block or subtask, calculations can begin in parallel if the boundary conditions of these blocks are known. Boundary conditions are sent by the processor element using message passing interface functions. The potential $U_{i,j}$ is updated repeatedly (eq.(10)). In addition, to the potential, the residual is also updated repeatedly (eq.(9)). To compute the potential at node i,j, the potential from the neighboring nodes are needed to be transferred. As an example, to compute potential at node i,j at *k+1* step :

send $U_{i,j}^{k}$ to all neighbors
(using MPI_Send function)
receive $U_{i+1,j}^{k}, U_{i-1,j}^{k}, U_{i,j+1}^{k}, U_{i,j-1}^{k}$ from neighbors
(using MPI_Rec function)
Compute $R_{i,j}$ using eq.(9)
Compute $U_{i,j}^{k+1}$ using eq.(10)

As shown from the above computations, for each potential updated at a boundary node, an additional time is added to the computation delay. This time is increased with increasing the number of blocks (processor element), increasing tolerant and increasing the number of nodes in the domain. So the division of the data domain into sub blocks is limited by the communication time.

CONCLUSIONS

Wireless sensor network have many limitations. One of these important limitations is the power consumption. This work suggested a method to reduce this power. This work is based on the analogy of the routing problem to the distribution of electrical field in a physical media with a given density of charges. From this analogy a set of partial differential equation is obtained. A finite difference method is utilized to solve this set numerically. Then, a parallel implementation is presented. The parallel implementation is based on domain decomposition, where the original calculation domain is decomposed into several blocks, each of which is given to a processing element. All nodes then execute computations in parallel, each node on its associated sub-domain. The time to compute routing in the network is reduced with increasing the number of the nodes (communication time is ignored). Of course the power consumption will decrease when the time is decreased.

REFRENCES

Andreas Adelmann1, Peter Arbenz, and Yves Ineichen1;"Improvements of a Fast Parallel Poisson Solver on Irregular Domains ".July 2010.

Azali Saudi1 and Jumat Sulaiman"Path Planning for Mobile Robot with Half-Sweep Successive Over Relaxation Iterative Method".Symposium on Progress in Information & Communication Technology 2009.

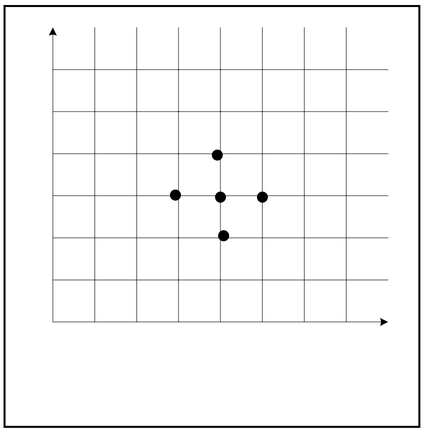
Mathew N.O Sadiku,"Numarical Techniques in Electromagnatics", second eddition, 2001.

M. Kalantari and M. Shayman, "Routing in Multi-Commodity Sensor Networks Based on Partial Differential Equations" IEEE ,2006, pp.402-406.

M. Kalantari and M. Shayman, "Routing in Wireless Ad Hoc Networks by Analogy to Electrostatic Theory" IEEE, 2004,pp. 4028-4033.

Vijay K. Garg "Wireless Communictions and Networking"© 2007 by Elsevier.

http://www.mhpcc.edu/training/workshop/mpi/Main.html parallel Programming Workshop_ Message Passing Interface.



Y

Figure 1 Finite difference mesh for two independent variables X and Y

I,j+1

I-1,j

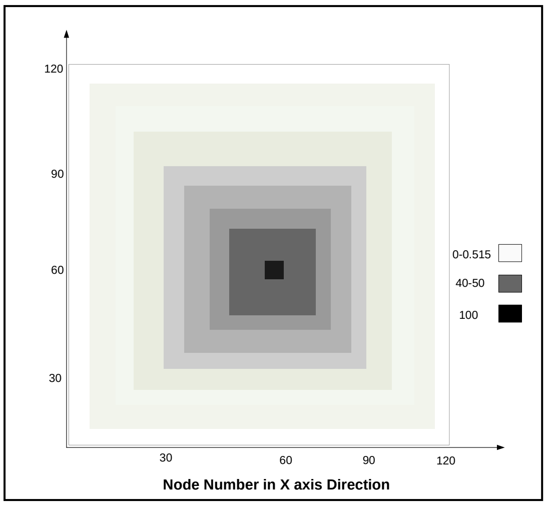


Figure.2 The value of the potential function *U*

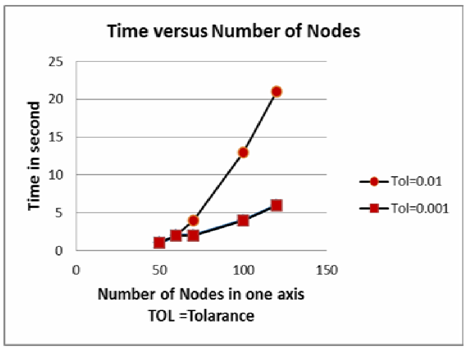


Figure 3 Execution time obtained with different mesh size and different tolerance.

umber in Y axis Direction

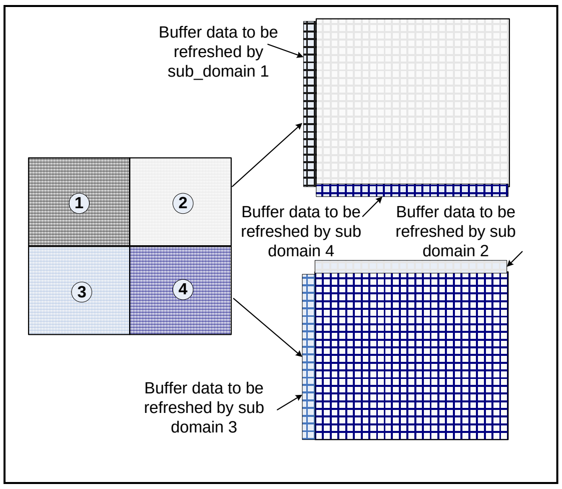

Buffer data to be refreshed by sub_domain 1
1
2
3
4
Buffer data to be refreshed by sub domain 4
Buffer data to be refreshed by sub domain 2
Buffer data to be refreshed by sub domain 3

Figure 4 Schematic representation of a calculation sub-domain divided into four subdomains, indicating the buffer cells used to store the internal boundaries data

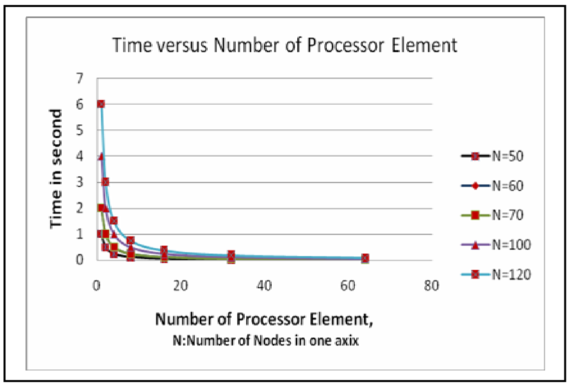


Figure 5.Execution time for different meshes size (50x50, 60x60, 70x70, 100x100, 120x120) with different number of processing element.

Table 1. Execution time for different meshes size (N number of nodes in one axis) with different number of processing element neglegting the communication time.

No of Processing elements	N=50	N=60	N=70	N-=100	N=120
1	1	2	2	4	6
2	0.5	1	1	2	3
4	0.24	0.5	0.5	1	1.5
8	0.12	0.25	0.25	0.5	0.75
16	0.06	0.125	0.125	0.25	0.375
32	0.03	0.0625	0.0625	0.125	0.1875
64	0.015	0.03125	0.03125	0.0625	0.09375