

# Protocols for Sensor Networks using COSMOS Model

Zhenyu Xu<sup>1</sup> and Pradip K Srimani<sup>1</sup>

Department of Computer Science, Clemson University, Clemson, SC 29634-0974

**Abstract.** Authors in [1] have recently introduced an interesting model, COSMOS (Cluster-based heterOgeneouS MOdel for Sensor networks) for sensor networks; COSMOS is a hierarchical network architecture that consists of a large number of low cost sensors with very limited computation capability and a smaller number of more powerful “clusterheads”. The clusterheads can communicate between each other in an asynchronous fashion while the low capability sensors under each clusterhead operate in a synchronous way with their respective clusterheads. Our purpose in the present paper is to design several protocols for benchmark programs like broadcast, matrix multiplication and matrix chain multiplication using this model and provide detailed complexity analysis of these protocols. Our results further illustrates the usefulness of the model for use in sensor networks.

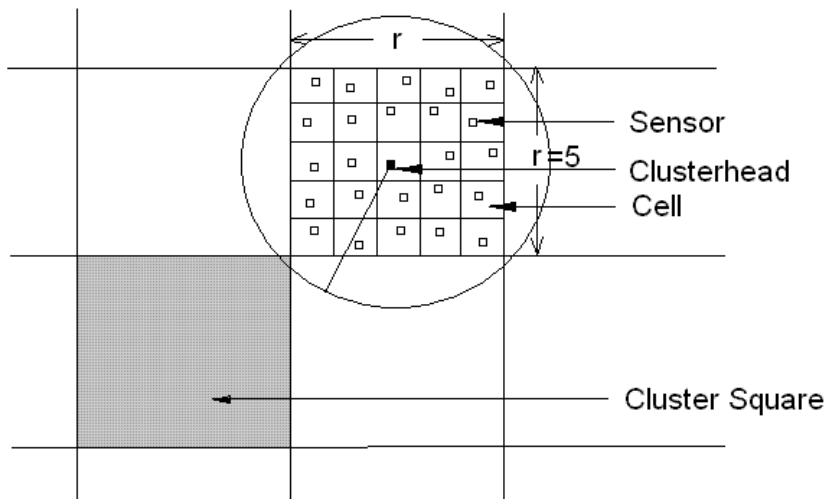
## 1 Introduction

Wireless sensor networks [2–5] consist of large number of tiny low-cost sensors that are used to sense natural phenomenon. These sensors have limited computation power as well as limited communication capability. We need specialized computing and communication protocols that can effectively adapt to these limitations of the sensor nodes.

Authors in [1] have recently introduced an interesting model, COSMOS (Cluster-based heterOgeneouS MOdel for Sensor networks) for sensor networks; COSMOS is a hierarchical network architecture that consists of a large number of low cost sensors with very limited computation capability and a smaller number of more powerful “clusterheads”. The clusterheads can communicate between each other in an asynchronous fashion while the low capability sensors under each clusterhead operate in a synchronous way with their respective clusterheads. Our purpose in the present paper is to design several protocols for benchmark programs like broadcast, and matrix multiplication using this model and provide detailed complexity analysis of these protocols.

## 2 The COSMOS Model

COSMOS model has been introduced in details in [1]. COSMOS assumes that the sensors are uniformly distributed in a two dimensional plane. The total area



**Fig. 1.** Clusters in COSMOS

is arranged as a grid of cells where each sensor occupies a cell. The sensors are organized into clusters, each cluster with a clusterhead which has a broader transmission range and more computational power than individual sensors. Within a cluster, the communication is single hop and its size is determined by the transmission range of the sensor. We assume the size of each cluster is  $r \times r$ , where the transmission range of the sensor is at least  $r/\sqrt{2}$ , as shown in Figure 1. The concept of clustering the sensors can also be applied in arbitrary networks [6]. However, the properties of particular topology such as mesh is utilized to simplify the computation and communication, as was done previously in [7] (the model was a strict arrangement of sensors in a mesh).

We assume the clusterhead knows the size of the sensor network and its own position (column and row index) in the mesh network. Each sensor has unit memory, unit processing power, and unit bandwidth. Each clusterhead has  $m \geq r^2$  memory,  $c \geq r^2$  processing power and  $b \geq r^2$  bandwidth. This enables the clusterhead to transmit or receive  $b$  data elements in one time step, either from other clusterheads or from the sensors within its cluster. All sensors in a cluster are time synchronized with their clusterhead. The communication between clusterheads is asynchronous using message passing.

## 2.1 Performance Metrics

To evaluate the proposed algorithms using the COSMOS computational model, we use three metrics of performance: Time complexity, Energy dissipation, and

Message complexity. These metrics were introduced in [1]; we briefly describe them in the following:

**Definition 1.** *Time complexity of an algorithm in the COSMOS model is defined to be the total execution time of the longest weighted execution chain on the clusterheads and sensors in the network.*

Time complexity includes the time taken to transmit, receive, or locally calculate data on clusterheads and sensors. The unit of data is the smallest data item on which computation or communication is performed. Since a clusterhead is more powerful in terms of computing power and bandwidth than a sensor, computation and communication at clusterheads are assigned higher weights. Each computation and communication of one unit of data at a sensor node is normalized to unity. The computation of one unit of data at a clusterhead is assigned a weight of  $1/c$  (a clusterhead is  $c$  times computationally more powerful than a sensor). Similarly, communication of one unit of data at a clusterhead is assigned a weight of  $1/b$  (a clusterhead has  $b$  times more bandwidth than that of a sensor).

**Definition 2.** *Total energy dissipation of an algorithm is defined to be the sum of energy consumed at sensors and clusterheads.*

We define the energy used to transmit, receive, or locally compute on one unit of data to be one unit of energy. [This assumes that the size of the sensor network small; the transmission energy is dominated by a range independent constant.

**Definition 3.** *Message complexity of an algorithm is defined to be the total number of messages transmitted in the execution of algorithm.*

A sensor always transmits and receives one unit of data in one message, since it has only one unit of memory. The message transmitted between clusterheads may contain multiple units of data.

## 2.2 System Primitives

We assume a underlying protocol provides reliable message passing between the sensors and clusterheads. Following system primitives are provided by the underlying protocol.

- **send** ( $i, j, x$ ) . The send primitive transmits the data  $x$  from the current clusterhead to another clusterhead labeled  $S_{i,j}$  within the transmission range. Both clusterheads maintains a local variable  $x$ . By calling this system primitive, the current clusterhead sends a message that contains the data in its local variable  $x$ . This message is received by clusterhead  $S_{i,j}$ , and  $S_{i,j}$  stores the data in its own local variable  $x$ .

It is apparent that the execution time of **send**( $i, j, x$ ) is  $|x|/b$ , where  $1/b$  is the weight of transmitting one unit of data between clusterheads, and the  $|x|$  is the size (number of units) of the data to be sent; and, the energy consumed in this process is  $|x|$ .

- `call (i, j, proc(args_list))` . This is a system primitive of RPC (remote procedure call). By calling this system primitive, the current clusterhead sends a message to a neighboring clusterhead  $S_{i,j}$ , indicating that  $S_{i,j}$  will invoke the local procedure  $proc$  with parameters  $args\_list$ . We assume the RPC message is short enough to be treated as one unit of data. Thus the execution time of this system primitive is  $1/b$ , and the energy consumed in this process is 1. It is possible to use different frequency to transfer data messages and RPC messages. In this case, there will be no collision between the two types of messages. In this paper, we assume only one frequency is used to transfer both types of messages so that only one clusterhead can be sending at the same time in the neighborhood of a particular clusterhead, no matter what type of the message to be sent.
- `wait (t)` . This system primitive simply let the clusterhead wait  $t$  units of time, without doing anything.

Throughout the paper, we use the notations shown in Table 1.

Symbol	Description
$n$	number of sensors in network
$S$	clusterhead
$r$	number of rows and columns of sensors in each cluster
$b, c$	weight of computation and communication cost on clusterhead
$m_1, m_2$	number of rows and columns of clusters in network
$a, b$	the row and column index of some particular cluster
$s, t$	the row and column index of some particular cluster
$i, j, k$	iteration index of row and column of clusters

Table 1. Notations

### 3 One to All Data Broadcasting

Consider a two dimensional  $m_1 \times m_2$  mesh of clusterheads, where  $S_{a,b}$  denotes the specific clusterhead,  $1 \leq a \leq m_1, 1 \leq b \leq m_2$ . A clusterhead  $S_{a,b}$  has some local data  $x$ . This data item  $x$  can be of any type; typically, it may be an array of integers or it may have a size of  $r^2$  where the clusterhead collects data from all the  $r^2$  sensor nodes that are attached to this clusterhead.

Without loss of generality, let the type of  $x$  be an array of integers. If only a single unit of data is to be broadcasted, the size of  $x$  is 1. Otherwise if more than one unit of data are to be broadcasted, the size of  $x$  is the number of data units. For example, when broadcasting the information collected from all the sensors attached to  $S_{a,b}$ , the size of  $x$  is  $r^2$ .

The COSMOS model does not include multicast as a feature, which can be used to flood the data from one clusterhead to all neighboring clusterheads in one step. Because multicast is not available in the network, we have to deploy strategies to minimize the time and energy needed. The most important issue in data broadcasting is the message collision. To prevent the message collision, there can be only one clusterhead sending the data at the same time, within the neighborhood of any clusterhead.

Consider the data broadcasting in a row of clusterheads. Assume the clusterheads are labeled  $S_{0,0}, S_{0,1}, S_{0,2}, \dots, S_{0,m}$ , and  $S_{0,0}$  contains the original data. In the first round,  $S_{0,0}$  sends data and RPC to  $S_{0,1}$ , and in the second round,  $S_{0,1}$  send the data and RPC to  $S_{0,2}$ . In  $m - 1$  rounds, all the clusterheads will get the data. Now consider these clusterheads will further send the data to the other nodes in the same column. Since  $R = r$ ,  $S_{0,i}$  and  $S_{0,i+1}$  can send messages to  $S_{1,i}$  and  $S_{1,i+1}$  relatively in the same time, without incurring collision. So the strategy is, first send the data to all the clusterheads in the same row, then these clusterheads send to all other clusterheads in the same column.

### 3.1 Algorithm

The pseudo code for the data broadcasting algorithm, **Broadcast(a, b, x)**, is shown in Figure 2. This algorithm broadcasts data  $x$  from the clusterhead  $S_{a,b}$  to all the clusterheads in the network, where  $1 \leq a \leq m_1, 1 \leq b \leq m_2$  and  $m_1 \times m_2$  are the size of the mesh of clusterheads. Before the algorithm executes, only  $S_{a,b}$  has the data  $x$ . When the algorithm ends, all the clusterheads have a local copy of  $x$ .

The algorithm **Broadcast(a, b, x)** has three parameters. Parameters  $a$  and  $b$  are the coordinates of the clusterhead that contains the data to be broadcast. Parameter  $x$  is the data.

We use the first parameter  $a$  to denote the row coordinate and the second parameter  $b$  to denote the column coordinate of the clusterhead. The coordinates are integers that are known to all the clusterheads. Thus when we say “all clusterheads on row  $a$ ”, we refer to all clusterheads of the form  $S_{a,j}$ , where  $1 \leq j \leq m_2$ . We use this naming convention in the remainder of this paper.

**Broadcast(a, b, x)** uses two subroutines: ColBroadcast(a, b, x), which sends the data  $x$  to a column, and RowBroadcast(a, b, x), which sends  $x$  to a row. Initially, Broadcast(a, b, x) is called on  $S_{a,b}$ , which contains the data  $x$ .

### 3.2 Time complexity

The algorithm can be divided into two phases. In the first phase, the data is sent to all clusterheads on row  $a$ . The clusterheads that get the data wait until data reaches all clusterheads on row  $a$ . After that, phase 2 starts and the data is sent along the columns.

**Theorem 1.** *In a  $m_1 \times m_2$  mesh of clusterheads that contains  $n$  sensors, the time complexity of **Broadcast(a, b, x)** is  $O(\sqrt{n})$ .*

*Proof.* In **RowBroadcast**, each clusterhead takes  $3/c$  units of time to do the comparison,  $|x|/b$  units of time to transmit the data and  $1/b$  unit of time to perform the RPC, then it starts waiting. So the execution time of RowBroadcast is  $\max(m_2 - b, b) \times (|x|/b + 1/b + 3/c)$ . The upper bound is  $m_2(|x|/b + 1/b + 3/c)$ .

Similarly, the execution time of ColBroadcast is  $m_1(|x|/b + 1/b + 2/c)$ . So the total execution time is  $m_2(|x|/b + 1/b + 3/c) + m_1(|x|/b + 1/b + 2/c)$ , which

Following code is executed on cluster  $S_{i,j}$ :

```

RowBroadcast(a, b, x)
Begin
    if  $j \geq b \wedge j < m_2 \wedge i = a$  then
        send(i, j+1, x)
        call(i, j+1, RowBroadcast(a,b,x))
    if  $j \leq b \wedge j > 0 \wedge i = a$  then
        send(i, j-1, x)
        call(i, j-1, RowBroadcast(a,b,x))
    if  $j > b \wedge j \leq m_2 \wedge i = a$  then
        wait( $\max(m_2 - j, 2b - j + 1)$ )
    else if  $i = a$  then
        wait( $\max(m_2 - (2b - j + 1), j)$ )
    else
        wait( $\max(m_2 - b, b - 1)$ )
End

```

```

ColBroadcast(a, b, x)
Begin
    if  $i \geq a \wedge i < m_i$  then
        send(i+1, j, x)
        call(i+1, j, ColBroadcast(a,b,x))
    if  $i \leq a \wedge i > 0$  then
        send(i-1, j, x)
        call(i-1, j, ColBroadcast(a,b,x))
End

```

Following code is executed on cluster  $S_{a,b}$ , which contains the original data to be broadcast:

```

Broadcast(x)
Begin
    RowBroadcast(a, b, x)
    ColBroadCast(a, b, x)
End

```

**Fig. 2.** Algorithm 2: One to All Broadcast Algorithm

is  $O(m_1 + m_2)$ . For a  $\sqrt{n}/r \times \sqrt{n}/r$  mesh,  $m_1 = m_2 = \sqrt{n}/r$ , time complexity is  $O(2\sqrt{n}/r) = O(\sqrt{n})$ .

### 3.3 Energy Dissipation

**Theorem 2.** *In a  $m_1 \times m_2$  mesh of clusterheads that contains  $n$  sensors, the energy dissipation of **Broadcast**( $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{x}$ ) is  $O(n)$ .*

*Proof.* Each clusterhead receives one data message and one RPC message, except for clusterhead  $S_{a,b}$ . So total number of data or RPC messages sent is  $\frac{n}{r^2} - 1$ . Each data message contains  $|x|$  units of data, and each RPC message contains 1 unit of data, so the energy dissipation for transmitting messages is  $(\frac{n}{r^2} - 1) \times |x| = O(n)$ .

In RowBroadcast, the number of comparisons performed on each clusterhead is 3. In ColBroadcast, the number of comparisons performed on each clusterhead is 2. So the total number of computations is  $3 \times m_2 + 2 \times m_1 \times m_2$ . Each computation on clusterhead takes 1 unit of energy. So the energy dissipation for computation is  $O(m_1 \times m_2) = O(n)$ .

### 3.4 Message Complexity

**Theorem 3.** *In a  $m_1 \times m_2$  mesh of clusterheads that contains  $n$  sensors, the message complexity of **Broadcast**( $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{x}$ ) is  $O(n)$ .*

*Proof.* Except for the initial clusterhead  $S_{a,b}$ , each clusterhead receives one data message and one RPC message. So total number of messages transmitted is  $\frac{2n}{r^2} - 2$ . Thus the message complexity is  $O(n)$ .

## 4 All to All Data Broadcasting

The All to All data broadcasting in COSMOS model is defined as all the clusterheads transmits data to every other clusterheads. It is possible to implement the all-to all data broadcasting by repeating the One to All data broadcasting  $m_1 \times m_2$  times. However, this approach is not time efficient. Two non-interfering clusterheads can be scheduled to transmit different data at the time to save execution time.

### 4.1 Data Structures and Algorithm

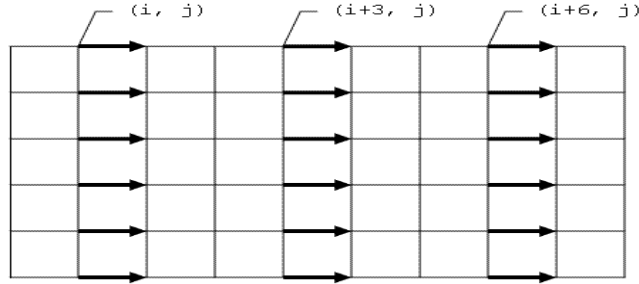
As in One to All data broadcasting, we assume each clusterhead maintains an integer variable  $x$  that contains the data to be broadcast to all other clusterheads.

Furthermore, to store the data comes from other clusterheads, each clusterhead also maintains an integer array  $Y[1..m_1][1..m_2]$  of size  $m_1 \times m_2$ . For each clusterhead  $S_{i,j}$ , we define a procedure **sync**( $\alpha$ ,  $\beta$ ), where the parameters  $\alpha$  and  $\beta$  can take values as shown in Table 2.

Consider all the clusterheads on row  $i$ . To prevent the collision, when  $S_{i,j}$  is executing **sync**(0, 1),  $S_{i,j+1}$  and  $S_{i,j+2}$  cannot execute **sync**(0, 1). However,

$\alpha$	$\beta$	Definition	Description
1	0	$\text{send}(i+1, j, Y[1..i][1..m_2])$	sends the upper part (up to row $i$ ) of $y$ to the lower neighbor of clusterhead $S_{i,j}$ .
-1	0	$\text{send}(i-1, j, Y[i..m_1][1..m_2])$	sends the lower part (up to row $i$ ) of $y$ to the upper neighbor of clusterhead $S_{i,j}$ .
0	1	$\text{send}(i, j+1, Y[1..m_1][1..j])$	sends the left part (up to column $j$ ) of $y$ to the right neighbor of clusterhead $S_{i,j}$ .
0	-1	$\text{send}(i, j-1, Y[1..m_1][j..m_2])$	sends the right part (up to column $j$ ) of $y$ to the left neighbor of clusterhead $S_{i,j}$ .

**Table 2.** The  $\text{sync}(\alpha, \beta)$  procedure



**Fig. 3.** Executing  $\text{sync}(0, 1)$  every three columns. Arrows denote the data transmission.

$S_{i,j+3}$  can execute  $\text{sync}(0, 1)$ , as well as other clusterheads in the same column. This is shown in figure 3.

In the first and second round, all clusterheads on column  $j, j+3, j+6, \dots$  execute  $\text{sync}(0, 1)$  and  $\text{sync}(0, -1)$ . This sends the data on those columns to adjacent columns. In the third and fourth round, all clusterheads on column  $j+1, j+4, j+7, \dots$  execute  $\text{sync}(0, 1)$  and  $\text{sync}(0, -1)$ . In the fifth and sixth round, all clusterheads on column  $j+2, j+5, j+8, \dots$  execute  $\text{sync}(0, 1)$  and  $\text{sync}(0, -1)$ . After the six rounds, each clusterhead contains the correct data value from its left and right neighbors.

This process is repeated  $m_2/3+1$  times. After these rounds, each clusterhead contains all the data from the clusterheads on the same row. Then all clusterheads execute  $\text{sync}(1, 0)$  and  $(-1, 0)$ , in the same way of every three rows, to transfer the data to the entire mesh. The formal algorithm is presented in figure 4

## 4.2 Time complexity

**Theorem 4.** In a  $m_1 \times m_2$  mesh of clusterheads that contains  $n$  sensors, the time complexity of  $\text{All2AllBroadcast}(\mathbf{x})$  is  $O(n^{3/2})$ .

```

Following code is executed on cluster  $S_{i,j}$ :

All2AllBroadcast(x)
Begin
     $Y[i][j] = x$ 
    for  $k = 0$  to  $m_2/3$  do
        {
            wait( $j \bmod 3$ )
            sync(0, 1)
            sync(0, -1)
            wait( $2 - (j \bmod 3)$ )
        }
    for  $k = 0$  to  $m_1/3$  do
        {
            wait( $i \bmod 3$ )
            sync(1, 0)
            sync(-1, 0)
            wait( $2 - (i \bmod 3)$ )
        }
End

```

**Fig. 4.** All to All Broadcast Algorithm

*Proof.* In the first loop of **All2AllBroadcast**, each clusterhead executes  $m_2/3 + 1$  times of **sync**(0, 1) and  $m_2/3 + 1$  times of **sync**(0, -1). In **sync**(0, 1),  $|x| \times m_1 \times (j + 1)$  units of data are transmitted. In **sync**(0, -1),  $|x| \times m_1 \times (m_2 - j)$  units of data are transmitted. So the execution time in the first loop is  $|x| \times m_1 \times (m_2 + 1) \times (m_2/3 + 1) \times 1/b = O(m_1 \times m_2^2)$ .

Similarly, in the second loop, the execution time is  $|x| \times (m_1 + 1) \times m_2 \times (m_1/3 + 1) \times 1/b = O(m_1^2 \times m_2)$ .

For a  $\sqrt{n}/r \times \sqrt{n}/r$  mesh,  $m_1 = m_2 = \sqrt{n}/r$ , time complexity is  $O(m_1 \times m_2^2 + m_1^2 \times m_2) = O(n^{3/2})$ .

Recall that the time complexity of One to All broadcast is  $O(n^{1/2})$ . If simply apply  $m_1 \times m_2$  times One to All broadcast on each clusterhead, the time complexity will be  $O(n^{1/2} \times n^2) = O(n^{5/2})$ . So algorithm **All2AllBroadcast** is more time efficient.

### 4.3 Energy Dissipation

**Theorem 5.** In a  $m_1 \times m_2$  mesh of clusterheads that contains  $n$  sensors, the energy dissipation of **All2AllBroadcast(x)** is  $O(n^{3/2})$ .

*Proof.* In the first loop of **All2AllBroadcast**, each clusterhead executes  $m_2/3 + 1$  times of **sync**(0, 1) and  $m_2/3 + 1$  times of **sync**(0, -1). In **sync**(0, 1),  $|x| \times m_1 \times (j + 1)$  units of data are transmitted. In **sync**(0, -1),  $|x| \times m_1 \times (m_2 - j)$  units of data are transmitted. So the energy dissipation in the first loop is  $|x| \times m_1 \times (m_2 + 1) \times (m_2/3 + 1) = O(m_1 \times m_2^2)$ .

Similarly, in the second loop, the energy dissipation is  $|x| \times (m_1 + 1) \times m_2 \times (m_1/3 + 1) = O(m_1^2 \times m_2)$ .

For a  $\sqrt{n}/r \times \sqrt{n}/r$  mesh,  $m_1 = m_2 = \sqrt{n}/r$ , total energy dissipation is  $O(m_1 \times m_2^2 + m_1^2 \times m_2) = O(n^{3/2})$ .

#### 4.4 Message Complexity

**Theorem 6.** *In a  $m_1 \times m_2$  mesh of clusterheads that contains  $n$  sensors, the message complexity of **All2AllBroadcast**( $x$ ) is  $O(\sqrt{n})$ .*

*Proof.* In the first loop of **All2AllBroadcast**, each clusterhead executes  $m_2/3 + 1$  times of `sync(0, 1)` and  $m_2/3 + 1$  times of `sync(0, -1)`. So the number of messages transmitted is  $m_2 \times 2/3 + 2$ . Similarly, in the second loop, the number of messages transmitted is  $m_1 \times 2/3 + 2$ .

For a  $\sqrt{n}/r \times \sqrt{n}/r$  mesh,  $m_1 = m_2 = \sqrt{n}/r$ , So total number of messages transmitted is  $\sqrt{n}/r \times 4/3 + 4$ . Thus the message complexity is  $O(\sqrt{n})$ .

## 5 Matrix Multiplication

Given two matrices  $A_{m \times m}$  and  $B_{m \times m}$ , the matrix multiplication  $C = A \times B$  can be calculated as  $C_{ij} = \sum_{1 \leq k \leq m} A_{ik} B_{kj}$ , where  $1 \leq i, j \leq m$ . In the COSMOS model, the matrix multiplication does the following: if for all the sensors,  $A_{ij}$  and  $B_{ij}$  is stored on cluster row  $s$  column  $t$ , and inside the cluster the sensor on row  $p$  column  $q$ , where  $i = (s - 1)r + p$ ,  $j = (t - 1)r + q$ . Then after the matrix multiplication is done, the result  $C_{ij}$  is stored in the same way.

### 5.1 Data Structure

Each sensor keeps three integer variables  $a$ ,  $b$ , and  $c$ . Before the algorithm is started,  $a$  and  $b$  contain the corresponding element of matrix  $A$  and  $B$ . After the algorithm is finished,  $c$  contains the element of matrix  $C = AB$ . Each clusterhead keeps following variables: Integer arrays  $X[1..m][1..m]$  and  $Y[1..m][1..m]$  of size  $m \times m$ , which store elements of  $A$  and  $B$  that get from the sensors within the cluster and from other clusterheads. An integer array  $z[1..m][1..m]$  of size  $m \times m$ , which stores computed elements of the result matrix  $C$ . Two integer variables  $s$  and  $t$  that denote the index of the clusterhead in the mesh.

### 5.2 Algorithm

The first step is aggregating data  $a$  and  $b$  from the sensors to array  $x$  and  $y$  of its clusterhead. For the clusterhead at row  $s$  and column  $t$ , it stores all the  $a$  elements to  $X[(s - 1)r + 1..s \times r][(t - 1)r + 1..t \times r]$ , and stores all the  $b$  elements to  $Y[(s - 1)r + 1..s \times r][(t - 1)r + 1..t \times r]$ . The next step uses the All to All data broadcasting to send the block of  $A$  and  $B$  matrices to all the clusterheads. The clusterhead  $S_{s,t}$  can then calculate the block  $z[(s - 1)r + 1..s \times r][(t - 1)r + 1..t \times r]$  as follows:  $z[i][j] = \sum_{0 \leq k < m} X[i][k] \times Y[k][j]$ . Finally,  $z[i][j]$  is distributed to the sensor on row  $i$  column  $j$  in the cluster.

Following code is executed on cluster  $S_{s,t}$ :

```

MM(a, b)
Begin
    do DataAggregation( $a, X[(s-1)r+1..s \times r][(t-1)r+1..t \times r]$ )
    do DataAggregation( $b, Y[(s-1)r+1..s \times r][(t-1)r+1..t \times r]$ )
    do All2AllBroadcast( $x$ )
    do All2AllBroadcast( $y$ )
    calculate  $z[(s-1)r+1..s \times r][(t-1)r+1..t \times r]$  using data in  $X[1..m][1..m]$ 
    and  $Y[1..m][1..m]$ .
    send result in  $z[(s-1)r+1..s \times r][(t-1)r+1..t \times r]$  back to sensors.
End

```

**Fig. 5.** Matrix Multiplication Algorithm

### 5.3 Time complexity

**Lemma 1.** *DataAggregation* takes  $8r^2 + 6$  time steps.

*Proof.* [1] gives the aggregation schedule: It takes  $2r^2$  time for one cluster to aggregate data: In the first  $r^2$  time, each sensor is assigned a rank order. In the next  $r^2$  time, sensors send data  $x$  to clusterhead in the rank order.

To avoid collision, adjacent clusters must not aggregate data in same period. The aggregation sequence is scheduled in this way: In time  $0 \leq t < 2r^2$ , all clusters on even row and even column do data aggregation. In time  $2r^2 + 2 \leq t < 4r^2 + 2$ , all clusters on even row and odd column do data aggregation. In time  $4r^2 + 4 \leq t < 6r^2 + 4$ , all clusters on odd row and odd column do data aggregation. In time  $6r^2 + 6 \leq t < 8r^2 + 6$ , all clusters on odd and even column do data aggregation. The 6 time steps are required to notify neighboring clusterheads to start the data aggregation. Thus the total time is  $8r^2 + 6$ .

**Theorem 7.** *In a  $m \times m$  mesh that contains  $n$  sensors, the time complexity of Matrix Multiplication is  $O(n^{5/2})$ .*

*Proof.* By lemma 1, data aggregation takes  $8r^2 + 6$  time steps. The All to All data broadcasting takes  $O(n^{5/2})$  time steps. The matrix multiplication is then concurrently performed on all the clusterheads. It takes  $2m/c$  time steps to compute one element in  $z$ , so in all it takes  $2r^2m/c$  time steps to compute all  $r \times r$  elements. The last step is the reverse of data aggregation, so it also takes  $8r^2 + 6$  time steps. Adding all together, the time complexity is  $O(n^{5/2})$ .

### 5.4 Energy dissipation

**Theorem 8.** *In a  $m \times m$  mesh that contains  $n$  sensors, the energy dissipation of Matrix Multiplication is  $O(n^{3/2})$ .*

*Proof.* The total number of data transmissions in data aggregation step is  $2n$ . The total number of data transmissions in last step (reverse data aggregation) is  $n$ . So the energy dissipation is  $O(n)$  for these steps. The energy dissipation in **All2AllBroadcast** is  $O(r^2) \times O(n^{3/2}) = O(n^{3/2})$ . Calculating  $z[1..r][1..r]$  in each cluster takes  $O(m \times r^2)$  calculations in all. Since  $r$  is fixed size of the cluster, it is a constant. So the energy dissipation on calculation is  $n \times O(m) = O(n^{3/2})$ . Therefore Total energy dissipation is  $O(n) + 2n^{3/2}/b + 2n = O(n)$ .

## 5.5 Message complexity

**Theorem 9.** *In a  $m_1 \times m_2$  mesh that contains  $n$  sensors, the message complexity of **Matrix Multiplication** is  $O(n)$ .*

*Proof.* The total number of transmissions in data aggregation and reverse data aggregation step is  $3n$ . Each of the **All2AllBroadcast** step sends  $O(n^{1/2})$  messages. So the total number of transmissions is  $O(n) + O(n^{1/2}) = O(n)$ .

## Acknowledgement

The work was supported by an NSF Award # ANI-0219485.

## References

1. M. Singh and V. K. Prasanna. A hierarchical model for distributed collaborative computation in wireless sensor networks. In *the International Parallel and Distributed Processing Symposium (IPDPS'03)*, 2003.
2. M. Tubaishat and S. Madria. Sensor networks: an overview. *IEEE Potentials*, 22(2):20–23, April 2003.
3. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Magazine*, August 2002.
4. K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, pages 16–27, October 2000.
5. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM MobiCom00*, pages 56–67, Boston, MA, 2000.
6. W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *IEEE Proceedings of the Hawaii International Conference on System Sciences*, pages 1–10, January 2000.
7. Loren Schwiebert, Sandeep K.S. Gupta, and Jennifer Weinmann. Research challenges in wireless networks of biomedical sensors. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 151–165, New York, NY, USA, 2001. ACM Press.