ISSN 1392–124X INFORMATION TECHNOLOGY AND CONTROL, 2012, Vol. 41, No. 4

Task Assignment and Migration in Wireless Sensor Networks via Task Decomposition

Heemin Park

Department of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea

Jae Won Lee *

School of Computer Science, Sungshin Women's University
34-da Bomun-ro, Seongbuk-gu, Seoul 136-742, Korea
e-mail: jwlee@sungshin.ac.kr

crossref http://dx.doi.org/10.5755/j01.itc.41.4.887

Abstract. Energy consumption of sensor networks are largely affected by task assignments to the nodes in the network. In this paper, a task assignment method to improve the performance of wireless sensor networks, which exploits task decomposition and transformation, is presented. The task assignment is formulated as an optimization problem by providing a cost function incorporating the task decomposition and transformation at the same time. To show feasibility of our proposed method, simulated annealing approach is adopted. We also provided a distributed task migration method to support run-time of the given task on the network. While executing tasks in a node, if the remaining energy is less than pre-defined threshold level, the tasks in the node will be migrated into a healthier neighbor node. The simulation results show that elaborate assignments and task decomposition can significantly improve performance of sensor networks.

Keywords: Wireless Sensor Networks, Task Decomposition, Task Allocation, Task Migration

1. Introduction

Many of the monitoring applications of Wireless Sensor Networks (WSN) such as fire detection, traffic monitoring and wildlife habitat monitoring often require aggregation (or fusion) tasks like average, summation, minimum and maximum. These aggregation functions can be performed at the base station node (or user node) after collecting data from all sensors or at the network nodes. The latter is preferred because it can save energy with less communications of which cost is higher than that of computation in wireless networks [1].

Tasks of monitoring applications can be described as DAGs (Directed Acyclic Graphs), or task graphs which consist of a set of nodes (i.e. tasks) and arcs representing data transmission paths. Energy consumption of WSN is determined by the amount of communications and computations which are decided by the structures of task graphs and each task's assignment to nodes in the network. So far, many researches have been done to optimize the tasks' assignment for improving WSN's performance. However, to our best knowledge, none of the previous works did try to change or modify the structure of task graphs even though it largely affects the performances of WSN.

The assignment problem itself of the task graphs to heterogeneous multi-processors system has been investigated in the area of distributed systems [2]. However, in despite of the expensive communication costs, the communication between nodes was not accounted seriously. In Bonfils and Bonnet [3] and Kumar et al. [4], they provided methods of task placement; however, their methods do not modify the structure of the given task graphs for performance improvement. There have been intensive researches on task assignments and scheduling for wireless sensor networks [5–8]. In Yu and Prasanna [5], energy balancing through clustering of task graphs was proposed; but, the authors focused on single hop networks and they also did not change the task graphs for optimization. Instead of assignments of tasks, Sensoy et al. [8] proposed resource allocation with distributed agent-based approach, and network topology was considered for task assignment and scheduling in [6]. However, neither of the two approaches does not modify task graphs for better performance. In [7], Zhigang *et. al* mentioned that sensing, computing, and communication tasks are decomposed into subtasks; however, they did not present the methods used for task decomposition and optimization. There is an other area in which decomposition and transformation of graph have been used; for example, techniques for high-level synthesis of VLSI (Very Large Scale Integrated Circuits) and digital signal processor used

the techniques [9] that we applied to task assignment to WSN.

In many cases of WSN applications, there are opportunities to improve performances by modifying the structures of the given task graphs while maintaining their functionalities. Therefore, we propose a way to decompose given tasks into sub-tasks and transform them for better performances in energy consumption. For instance, a four-input ADD operator can be decomposed into two smaller ADD operators and transformed into a tree as shown in Fig. 1 (c). Since the four-input ADD task was broken into two sub-tasks, there are more flexibility in assigning the task and the data transmissions could be done with shorter communications. The decomposed and transformed task graph must have the same functionality with that of the original one. How to deal with the task transformation and assignment at the same time is not a straightforward problem. For this, we formulate the task assignment as an optimization problem of which cost function includes energy consumption and latency while taking into account the task decomposition and transformations in the optimization framework.

In addition to the task assignment via task decomposition and transformation, we further investigated the following. One of important characteristics of WSN is that if some of nodes dies by depletion of battery, then the entire network would no longer be functional. Therefore, each node itself needs to decide if tasks inside the node are better to be transferred to other nodes to make the entire network survive longer. Since this task migration procedure is hard to be managed in central servers, there should be some distributed mechanism to support *run-time* of sensor networks. Even after depleting energy of some nodes which conducted computation and/or communication intensive jobs, there would be still healthy neighbor nodes nearby that have enough energy to continue doing the jobs. Then, if nodes of low energy could transfer their tasks to other healthier nodes, the lifetime of the network would be extended further. In Fig. 1 (d), it shows the case that when one of nodes of ADD task depletes its battery, the ADD task migrates to neighbor (right one) node and continues the task.

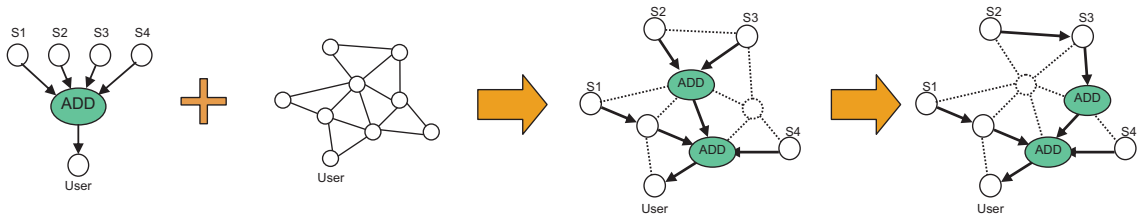
The remainder of this paper is organized as follows. In section 2, we present the problem formulation and Section 3 presents an implementation of proposed task assignment and migration. Experimental results using an illustrative example are shown in Section 4. Then, we conclude in Section 5.

2. Problem Formulation

A task for sensor networks is represented as a DAG (Directed Acyclic Graph) where nodes represent functions to be performed in the sensor nodes and arcs represent data transmissions between the nodes. One of the nodes serves as a user node which will store the final results. A task graph $G = (V, E)$ and a deployment of sensor nodes $S = \{s_i | i = 1, 2, ..., n\}$ are given, where n is the number of deployed sensor nodes. Let $V = \{v_i | i = 1, 2, ..., m\}$ be a set of tasks where m is the number of tasks in the task graph. There are arcs $(u, v) \in E$ iff data are transmitted from node u to node v. Then the task assignment problem for wireless sensor networks can be defined as following. The problem is to *find a task transformation (including decomposition)* $T : G \rightarrow G' = (V', E')$ *and a mapping (or assignment) of tasks* $A : V' \rightarrow S$ *to the sensor nodes such that the total cost is minimized*. The transformation *T* is a kind of synthesis from decomposable tasks to transformed task graphs. If the function of a task is commutative and associative, such as the summation, maximum, minimum and average, then the task can be decomposed into tree structures. This can be done by splitting an operator with large number of inputs into operators with smaller number of inputs. This decomposition process can be continued until the operator becomes *atomic*. The task which cannot be decomposed nor transformed is called an *atomic* task in this paper. Once the tasks are decomposed and transformed into tree structures, there are more opportunities to minimize the total energy consumption thanks to increased flexibility and shorter communications. When the tasks of large number of inputs are decomposed into several sub tasks with less number of inputs, the distance of communication links between tasks tend to be shorter than before. Then, there would be more chances to optimize the total energy consumption of the network by exploiting the fact that energy consumption of communication is much expensive than that of computation. Multi-hops with short-range transmissions are more energy efficient than single hops with long-range transmissions.

3. Implementations

We implemented the task assignment problem formulated in Section 2 with an optimization approach. For the simplicity of implementation, simulated annealing framework is adopted. Simulated annealing is a well-known iterative optimization method for combinatorial optimization problems [10]. The reason of choosing simulated annealing method is that it is easy to implement and it can accommodate



(a) Task Graph (b) Network Topology (c) Task Assignment on the Deployed Network (d) Task Migration

Figure 1. An Example of Task Decomposition, Transformation, and Migration: ADD Operator

various forms of combinations (e.g. modifications) in the solution space. For implementing simulated annealing, four things are required to be prepared: initial solution, cost function, neighborhood solutions, and temperature scheduling [10]. As a conventional way of temperature scheduling is used, cost function and neighborhood solutions are mainly explained in this section.

3.1. Cost Function

We propose a cost function that can accommodate decomposition, transformation, and assignment of the task graphs at the same time. The cost function should indicate which result is better when hopping different combinations of task assignments from the search space. We defined the cost function of transformed task graph G' and its assignment A as follows in order to accommodate various trade-offs:

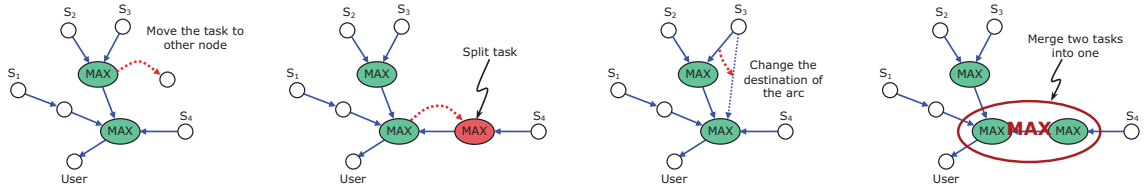
$$\begin{aligned} Cost(G', A) = & \sum_{v_i \in V'} E_{comp}(v_i) \\ & + \sum_{(u,v) \in E'} E_{comm}(u, v) \\ & + w_E \times \max_{v_i \in V'} \{E(v_i)\} \\ & + w_L \times L(G', A). \end{aligned} \quad (1)$$

In Equation (1), $E_{comp}(v_i)$ is the energy consumed for computation inside the sensor node v_i, $E_{comm}(u, v)$ energy consumed at u for transmitting data to v, $E(v_i)$ the energy consumption at sensor node v_i, and w_E and w_L are parameters to provide weights to maximum energy consumption and latency, respectively. The $L(G', A)$ is the maximum time delay between the user node to sensor nodes and $L(G', A)$ is defined to be the time of the longest path from all sensor nodes to the user node. It includes time for processing at each node and transmission of data between nodes. The total cost is the summation of energy consumption for communication and computation for all tasks, and weighted summation of latency and maximum energy consumption from all nodes. The meanings of w_E and w_L are weights to adjust the priorities of maximum energy consumption for each node and maximum latency of the assignment, respectively. Therefore, users just need to increase the weight that they want to set higher priority. In general, the life time of a network is the duration of the alive time of the node that dies first. Therefore, the energy consumption of the node that consumes the most is necessary to be minimized. Assigning large values to w_E can force to minimize $max_{v_i \in V'}\{E(v_i)\}$. When $max_{v_i \in V'}\{E(v_i)\}$ is supposed to be minimized, there would be many nodes which consume just less energy than $max_{v_i \in V'}(E(v_i))$. Therefore, the total energy consumption also has been included in the cost function. By adjusting w_E and w_L, users can adjust the cost function to meet the requirements of their applications.

All nodes are assumed to be time synchronized and have control knobs for shutdown and transmission power control. This means that each node will sleep when it is idle. Also, if the distance between nodes is short, the transmission power can be adjusted so as to reduce energy for communication. Another assumption is that program code of task is portable like Java and thus it can be ported to all the types of node in the network. Energy consumption for computation of task v_i, $E_{comp}(v_i)$, can be calculated as follows.

$$E_{comp}(i) = d_{in}(v_i) \times \frac{N(v_i)}{f(A(v_i))} \times I(A(v_i)) \quad (2)$$

where $d_{in}(v_i)$ is the number of arcs coming to task v_i (in-degree), $A(v_i)$ the sensor node at which task v_i is currently assigned, $N(v_i)$ the number of CPU cycles required to execute task v_i, $f(n)$ CPU frequency of node n, and $I(n)$ active CPU current per second in node n. The term $\frac{N(v_i)}{f(A(v_i))}$ means execution time of the task v_i. Thus, term $\frac{N(v_i)}{f(A(v_i))} \times I(A(v_i))$ represents the energy consumption for computation of task v_i at the sensor node $A(v_i)$. Because each node receives



(a) MOVE_TASK: move tasks from one to another node

(b) SPLIT_TASK: split tasks into two with same operators. One has data input and the other has data output

(c) MOVE_ARC: the destination of arcs can be moved to the adjacent tasks of the destination task.

(d) MERGE_TASK: two adjacent tasks can be merged into one

Figure 2. Neighborhood Solutions for Simulated Annealing

data only once from each sender node at a time in wireless networks, the task should be performed multiple times in order to receive multiple data from multiple senders; to accommodate this, $d_{in}(v_i)$ is multiplied. Likewise, energy consumption for communication is,

$$E_{comm}(u,v) = D_w(u) \times E_{trans}(A(u), A(v)) \quad (3)$$

where $(u,v) \in E'$, $D_w(u)$ is the word size of output data from task u, $E_{trans}(a,b)$ consumed energy to transmit 1-bit from sensor node a to sensor node b.

Energy consumption to transmit data between two nodes is dependent on the underlying routing protocol implemented in the network layer. If a specific routing protocol is adopted, it is enough to modify $E_{trans}(a,b)$ function according to the value of expected energy consumption to transmit data over routes between node a to b using that specific protocol. Although our method does not rely on any specific routing protocol, we assume that the routes of the minimal energy consumption are used when assigning tasks. The routes of minimal energy consumption for all pair of nodes are obtained by Dijkstra shortest path algorithm [11] in the beginning of the assignment program. For the nodes within radio ranges, the following general formula is used to calculate energy consumption for data transmission between sensor nodes a and b.

$$E_{trans}(a,b) = \alpha + \beta \times d(a,b)^{\gamma} \quad (4)$$

Here, $d(a,b)$ is the Euclidean distance between nodes a and b. The terms α and β are dependent on the radio hardware used in sensor nodes. The term γ represents path loss coefficient which is normally from 2 to 4. Since the cost function we propose can be modified for various types of node, our task assignment framework can be applied to various kinds of platforms.

3.2. Neighborhood Solutions

Simulated annealing is an iterative improvement method which selectively adopts neighborhoods of current solutions based on the probabilities [10]. We devised four kinds of neighborhood solutions that can cover task assignments, decomposition, and transformations at the same time, which are MOVE_TASK, SPLIT_TASK, MOVE_ARC and MERGE_TASK. SPLIT_TASK, MOVE_ARC and MERGE_TASK are used for task decomposition and transformations. The four neighborhood solutions and their brief explanations are shown in Fig. 2.

3.3. Distributed Task Migration

For run-time support of the sensor networks, a distributed task migration algorithm is developed. Basically, nodes that have low energy level have responsibilities to migrate their tasks to healthier neighbor nodes before they die. To guarantee for the nodes to have time before they die, the nodes start searching neighbors for transferring task when they reach a certain level of energy (e.g. 5% remaining energy). Pseudo code of the algorithm is shown in Fig. 3. As shown in Fig. 3 (a), the node that reached low energy level initiates task migration process by sending REQUEST_COST packets to its neighbors. Neighbors will respond with the cost when the tasks would be migrated to that node. After collecting all the costs from neighbors, the node can determine which neighbor node is the best for the tasks to be transferred. If the neighbors receive REQUEST_COST packet, they calculate cost when tasks are transferred to those nodes by requesting cost to incoming and outgoing tasks of the originator task. Thus, determining the node for task migration is done in two phases.

4. Experimental Results

4.1. Experimental Setup

For performance evaluation of our proposed approach, we have implemented the task assignment

```
if energy level < Threshold begin
    for each neighbor n begin
        Send REQUEST_COST packet to node n
        with the information of in/out tasks
    end

    Collect REPLY_COST packets
    Decide the best neighbor based on costs
    Transfer tasks to the chosen neighbor
end
```

(a) For the node of low energy level

```
if received REQUEST_COST packet begin
    if the packet does not come from
        the node of low energy level begin
        reply REPLY_COST packet
        exit
    end

    for each in/out task u begin
        Send REQUEST_COST packet to node u
    end

    Collect REPLY_COST packets
    Sum all the collected cost
    Reply REPLY_COST packet
end
```

(b) For the node that received REQUEST_COST packet

Figure 3. Task Migration Algorithm

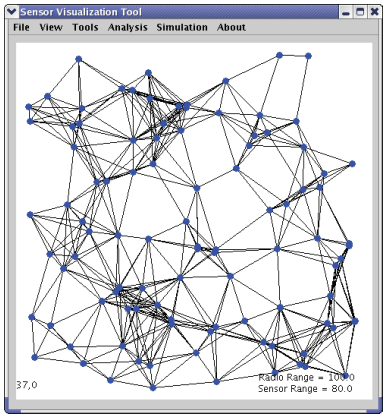


Figure 4. Deployment of 100 Sensors and Its Network Topology

with C language in Linux environment. Then we evaluated the distributed task migration method using Sensorsim [12]. The Sensorsim is an enhanced version of the ns-2 network simulator [13]. The aggregation functions and distributed task migration algorithm are implemented inside the sensor application layer.

To generate a field of sensor network, 100 sensor nodes are randomly deployed on $500m \times 500m$ area as shown in Fig. 4. Sensor node's maximum radio range is $100m$. The links between nodes represent that two nodes are within radio range each other. This means the communication is assumed to be symmetric.

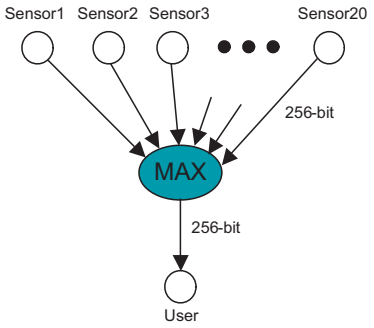


Figure 5. Task Graph of Maximum of 20 Nodes

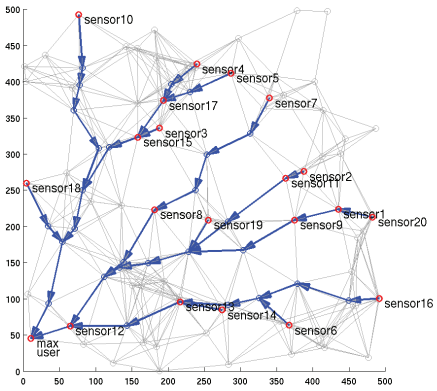
Table 1. Parameters for Estimating Energy Consumption

Item	Value
α	100 nJ/bit
β	0.07 nJ/bit/m^2
γ (path loss coefficient)	2
Batter depletion model	linear
Active CPU current in a node	42.23 mA
Current for radio transmission	79.14 mA
Current for radio reception	41.41mA

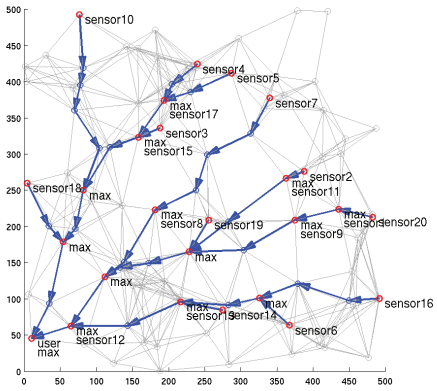
The sensor node used in the experiment is WINS node used in [14]. It equips 133MHz StrongARM S1100 processor and $100m$ range radio. Fig. 1 shows the parameters used in calculating energy consumption during task assignment. The term α includes the energy consumption in electronics part in radio, and also includes the energy consumption of CPU when forwarding packets.

4.2. Evaluation of Proposed Task Assignment

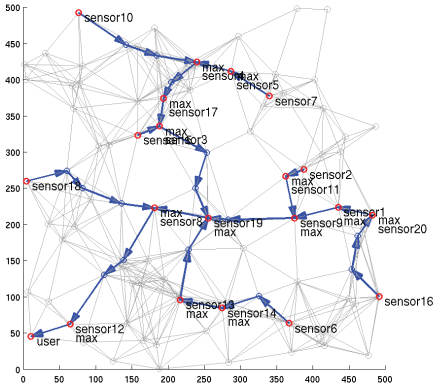
The example we used for evaluation of proposed task assignment is to get a maximum value from 20 sensors out of 100 deployed sensors. The task graph is shown in Fig. 5. Since MAX operator is commutative and also associative, it can be decomposed and transformed whenever needed. For experimental purpose, 20 sensors are randomly chosen and their locations are fixed. The node at the left lower corner serves as a user node. We compared the estimated energy consumption for four cases: initial assignment, assignment along minimal energy routes, optimized assignment for minimal energy consumption ($w_E = 0$ and $w_L = 0$), and optimized assignment for minimizing maximum energy consumption nodes ($w_E = 10$ and $w_L = 0$). Assignment results on the network topology and the estimated energy consumption of each option are shown in Fig. 6 and Table 2, respectively.



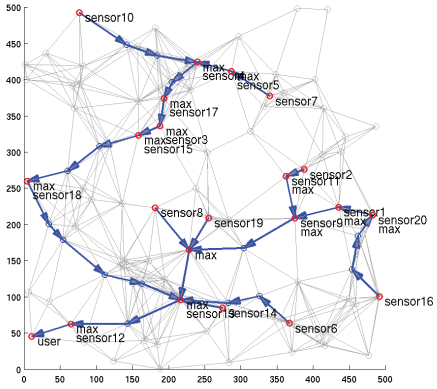
(a) Initial Task Assignment without Task Decomposition and Transformations



(b) Task Decomposition and Transformation along Minimal Energy Routes



(c) Optimized Task Assignment for Minimal Energy Consumption ($w_E = 0$ and $w_L = 0$)



(d) Optimized Task Assignment for Minimizing Maximum Energy Node ($w_E = 10$ and $w_L = 0$)

Figure 6. Results of Task Assignments

Table 2. Estimated Energy Consumption [uJ] for Each Assignment Options

Part of Energy Consumption	Initial Assignment	Assignment along Minimal Energy Routes	Optimized Assignments	
			$w_E = 0$ and $w_L = 0$	$w_E = 10$ and $w_L = 0$
Computation	6.4	10.5	10.2	10.2
Communication	14224.6	6807.7	4883.0	5205.6
Maximum	1184.4	205.4	178.3	147.8
Total	14231.0	6818.2	4893.1	6693.2

At the initial assignment, the task graph was not decomposed nor transformed. So, the MAX operations are assigned to the user node as shown in Fig. 6 (a). In this case of initial assignment, the user node collects data from each sensors and the MAX operation is performed at the user node. As mentioned in Section 3.1, communications between two nodes are assumed to be done through minimal energy routes.

Obviously, even though data transmissions were done through minimal energy routes, this naive task assignment would spend much more communication cost because every sensor node needs to send data to the user node through long routes, and there would be heavy traffic near the user node.

It is easy to notice that if we decompose the MAX operator and place them at every intermediate nodes which have more than one input arcs along minimal energy routes, the traffic would be reduced. This is a straightforward way of task decomposition and transformation as shown in Fig. 6 (b). We adopted this assignment as an initial task assignment (solution). However, this is still not the optimal task assignment for minimal energy consumption.

In Fig. 6 (c), the optimized task assignment for minimal energy consumption by our framework is shown. This result is obtained by setting w_E and w_L to zero. So, the objective of cost function is to minimize the total energy consumption of the entire network. When any of the network nodes die, the entire network may not work properly, therefore it is desirable for every node to have similar battery levels to extend the network lifetime. This can be achieved by minimizing the maximum of each node's energy consumption. If w_E is increased, we can achieve this purpose of minimizing the maximum energy consumption among nodes. In Fig. 6 (d), we get an optimized assignment to maximize the network lifetime by setting $w_E = 10$ and $w_L = 0$.

As shown in Table 2, the total energy consumption and nodes' maximum energy consumption could be significantly reduced by elaborate task assignments. The amount of consumed energy is for one sampling period, and it is assumed that dynamic shutdown control for CPU and radio, and power control for radio are available. The fourth column, "$w_E = 0$ and $w_L = 0$", shows that the total energy consumption was reduced by more than 65% and 28% from that of the initial assignment and the straightforward assignment, respectively. Moreover, as shown in the last column of the table, "$w_E = 10$ and $w_L = 0$", although the total energy consumption increased a little bit, but still the nodes' maximum energy consumption were reduced by more than 87% and 28% from those of the initial assignment and straightforward assignment, respectively.

4.3. Run-time Support: Distributed Task Migration

To demonstrate task migration feature, we made an artificial example that has atomic tasks and the output sizes are different according to the operations. The task is "beamforming" to calculate line of bearing (LOB) estimation from six sources. The task graph is shown in Fig. 7. We put *compress* task to reduce the output data size after FFT operations. Since beamforming operation (BEAM) and *compress* (COMP) task are computation intensive, the nodes at which BEAM and COMP are assigned would consume energy rapidly. The size of output data of sensors and output of FFT operation are 1024-bit, and 256-bit and 32-bit after COMP and BEAM tasks, respectively. In this case, all the tasks are atomic, and the sensors and user node have fixed locations. Because of the big size of the tasks, no more than two tasks can fit into a node. With our energy-efficient task assignment method, we get an optimized task assignment as shown in Fig. 8.

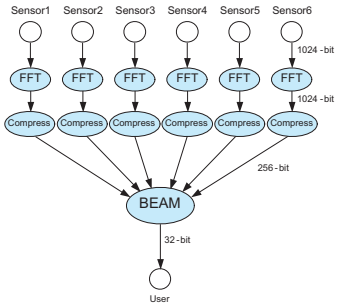


Figure 7. Task Graph of Beamforming

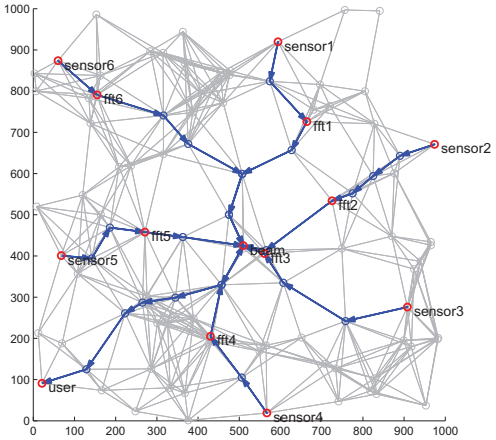


Figure 8. Optimized Assignments of Beamforming

The distributed task migration algorithm in Fig. 3 is implemented in sensor application layer of Sensorsim. A node will start task migration process if the energy of battery remains less than 5%. For the initial energy, we set 360 Joule for each node. The one sample period of the application is 10 second. When determining nodes for task migration, transmission range is not adjusted because each node should be able to communicate to any neighbors for cost calculations. In Fig. 9, the migration paths of BEAM task and COMP task are depicted. The BEAM task is as-

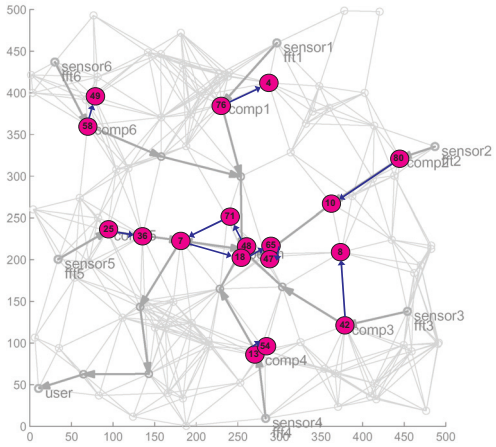


Figure 9. Migration Paths of BEAM and COMP Task

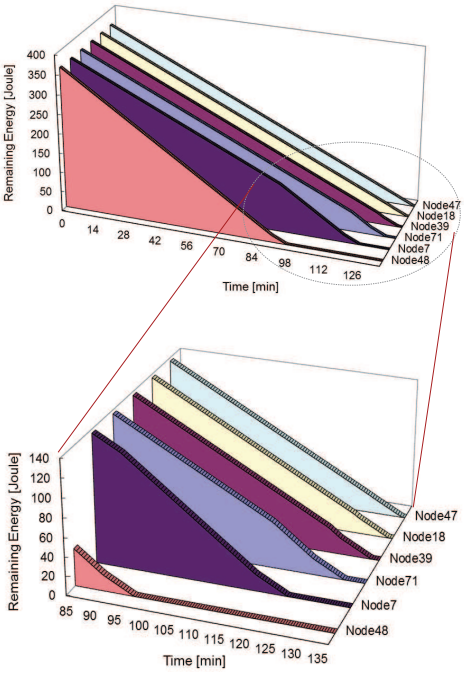


Figure 10. Remaining Energy in Nodes for Task Migration of BEAM

signed to node 48 at first and the battery of node 48 is depleted at 97.5 mins. At 91 mis, node 48 reached below of 5% of energy level and migration of tasks to other node is required before its battery is entirely depleted. BEAM task migrates from node 48 through 71, 7, 18, 65 47 as battery reaches threshold level (5%). Since COMP task is also computation-intensive task COMP tasks also migrate to their neighbor. Thanks to task migration feature, user task can get information until 136.5 mins which is 40% longer time than that without task migration feature. In Fig. 10, remaining energy of the nodes that BEAM task goes through are shown. As shown in Fig. 10, the battery in node 48 is consumed the first, then the battery in node 71 and node 7 and so on.

5. Conclusions

In this paper, task assignment for wireless sensor networks for minimal energy consumption has been presented. Task assignment should address not only communication and computation trade-offs, but it should also address task decomposition to enable flexible task assignments to network deployments. We propose a systematic approach to do task decomposition and transformation in conjunction with minimization of the total energy consumption. The proposed method formulates the problem in an optimization framework and shows improvements in estimation by more than 28% for total energy consumption and maximum energy consumption in all nodes. For run-time support, a distributed task migration has been suggested. A network simulation with an illustrative example showed that distributed migration of tasks at run-time can extended 40% longer the lifetime of network.

References

[1] **V. Raghunathan, C. Schurgers, S. Park, M. B. Srivastava.** Energy-Aware Wireless Microsensor Networks. *IEEE Signal Processing Magazine*, 2002, 19, pp. 40–50.

[2] **M. Kafil, I. Ahmad.** Optimal task assignment in heterogeneous distributed computing systems, *Concurrency, IEEE [see also IEEE Parallel & Distributed Technology]*, 1998, 6, pp. 42–50.

[3] **B. J. Bonfils, P. Bonnet.** Adaptive and Decentralized Operator Placement for In-Network Query Processing. *Information Processing in Sensor Networks (IPSN03)*, 2003, pp. 47–62.

[4] **R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, U. Ramachandran.** DFuse: A Framework for Distributed Data Fusion, *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 114–125.

[5] **Y. Yu, V. Prasanna.** Energy-Balanced Task Allocation for Collaborative Processing in Wireless Sensor Networks. *Mobile Networks and Applications*, 2005, 10, pp. 115–131.

[6] **B. Zhao, M. Wang, Z. Shao, J. Cao, K. C. C. Chan, J. Su.** Topology Aware Task Allocation and Scheduling for Real-Time Data Fusion Applications in Networked Embedded Sensor Systems. *14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA '08*, 2008, pp. 293–302.

[7] **L. Zhigang, L. Shining, Z. Xingshe, Y. Zhiyi.** Energy-efficient task allocation for data fusion in

Wireless Sensor Networks. *Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous*, 2009, pp. 1–6.

[8] **M. Sensoy, T. Le, W. W. Vasconcelos, T. J. Norman, A. D. Preece.** Resource Determination and Allocation in Sensor Networks: A Hybrid Approach. *The Computer Journal*, 2011, 54, pp. 356–372.

[9] **G. D. Micheli.** Synthesis and Optimization of Digital Circuits, *McGraw-Hill*, 1994.

[10] **S. Devadas, A. R. Newton.** Algorithms for hardware allocation in data path synthesis, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1989, 8, pp. 768–781.

[11] **T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein.** Introduction to Algorithms, *The MIT Press*, McGraw-Hill, 2001.

[12] **S. Park, A. Savvides, M. B. Srivastava.** Simulating networks of wireless sensors. *Winter Simulation Conference*, 2001, pp. 1330–1338.

[13] The Network Simulator - Ns-2, http://www.isi.edu/nsnam/ns/.

[14] **A. Savvides, S. Park, M. B. Srivastava.** On Modeling Networks of Wireless Micro Sensors. *SIGMETRICS 2001*, 2001, 318–319.

Received January 2011.