Comparison of Hyper-DAG Based Task Mapping and Scheduling Heuristics for Wireless Sensor Networks

Yuan Tian, Füsun Özgüner, and Eylem Ekici

Department of Electrical and Computer Engineering,
The Ohio State University
{tiany, ozguner, ekici}@ece.osu.edu

Abstract. In-network processing emerges as an approach to reduce energy consumption in Wireless Sensor Networks (WSN) by decreasing the overall transferred data volume. Parallel processing among sensors is a promising approach to provide the computation capacity required by in-network processing methods. In this paper, Hyper-DAG based Mapping and Scheduling (HDMS) algorithms for energy constrained WSNs are introduced. The design objective of these algorithms is to minimize schedule lengths subject to energy consumption constraints. Simulation results show that the CNPT-based HDMS algorithm outperforms other heuristic algorithms with respect to schedule lengths and heuristic execution times subject to energy consumption constraints.

1 Introduction

In-network processing is shown to be an effective approach to reduce energy consumption in Wireless Sensor Networks (WSN) by decreasing the overall transferred data volume [1]. In-network processing methods demand considerable computation capacity subject to timeliness requirements and energy consumption constraints. Sensors' computation power can be a bottleneck to process data given timeliness requirements. [1], [2], [3] propose collaborative data processing architectures by executing low level tasks on sensing sensors and offloading all other high level processing tasks to cluster heads. However, processing high level tasks can still exceed the capacity of cluster heads' computation power.

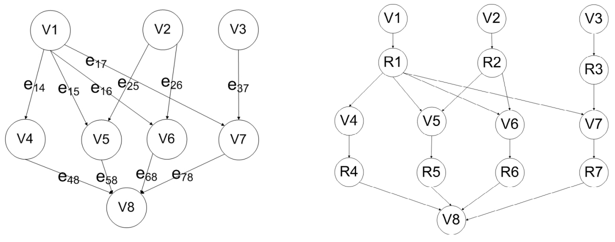
Parallel processing among sensors is a promising solution to provide the computation capacity required by in-network processing. Task mapping and scheduling has been extensively studied in the area of high performance computing [4] [5], but remains largely unexplored in WSNs. Task scheduling problem in WSNs has been considered in the literature recently. In [6], an online task scheduling mechanism (CoRAl) is presented to allocate the network resources between the tasks of periodic applications in WSNs. However, CoRAl does not address task mapping problem. Task mapping mechanisms in wireless networks have been presented in [7], [8]. Both [7] and [8] assume an existing underlying network communication mechanism without explicitly discussing communication scheduling between tasks. Task mapping and task scheduling have been considered jointly for mobile computing in [9]. However, the communication model adopted in [9] is not well suited for WSNs.

In this paper, three Hyper-DAG based Mapping and Scheduling (HDMS) algorithms for WSNs are introduced. The design objective of the algorithms is to minimize schedule lengths subject to energy consumption constraints. The performance of these algorithms is evaluated through simulations. Simulation results show that the CNPT [10] based HDMS algorithm outperforms the other heuristic algorithms with respect to schedule lengths and heuristic execution time subject to energy consumption constraints.

2 Network Assumptions and Task Mapping and Scheduling Problem

Our proposed HDMS algorithms are designed for applications executed within a cluster of homogeneous wireless sensor networks. The following assumptions are made for the sensor network: 1. Sensors are grouped into single-hop clusters with a clustering algorithm. Cluster heads create and coordinate schedules for communication and computation within clusters. 2. Sensors within a cluster are time synchronized. 3. Computation and communication can occur simultaneously on sensor nodes. 4. The communication within a cluster is isolated from other clusters through channel-hopping mechanisms.

An application executed in a cluster is modeled by a Directed Acyclic Graph (DAG) $T = (V,E)$, where the vertex set V denotes the tasks to be executed and the edge set E denotes the communication and dependency between the tasks. The computation cost of a task is represented by the number of CPU clock cycles to execute the task. The communication cost is denoted by the data volume to be transferred between tasks. Let $v_i \in V$ denote a task of the application and $e_{ij} \in E$ denote the connection between tasks v_i and v_j. v_i is the immediate predecessor of v_j and v_j is the immediate successor of v_i. A task without predecessors is called an *entry-task* and a task without successors is called an *exit-task*. A DAG may have multiple entry-tasks and one exit-task.



(a) A DAG Example (b) Hyper-DAG Extension

Fig. 1. DAG and Hyper-DAG Example

Fig. 1(a) shows a DAG example, where V1 is an entry-task, V8 is an exit-task, and V5 is the immediate successor and predecessor of V1 and V8, respectively.

Let $H^x = \{h_1^x, h_2^x, ..., h_n^x\}$ denote a set of mapping and scheduling decisions of an application T on a network G. Each element $h_i^x \in H^x$ is a tuple of the form $(v_i, m_k, s_{ik}, f_{ik}, c_{ik})$, where m_k represents the sensor to which task v_i is assigned, s_{ik} and f_{ik} represent the start time and finish time of v_i, and c_{ik} represents the energy consumption of v_i on node m_k, respectively. For convenience, let $pred(v_i)$ and $succ(v_i)$ denote the immediate predecessors and successors of task v_i respectively, $m(v_i)$ denote the sensor on which v_i is assigned, and $T(m_k)$ denote the tasks assigned on sensor m_k. The target problem P^o is to find an $H^o \in \{H^x\}$ that has the minimum schedule length under the energy consumption constraint. P^o is an NP-complete problem in general [5]. Therefore heuristic algorithms to solve P^o are needed.

3 Hyper-DAG Based Mapping and Scheduling Basics

The proposed HDMS algorithms are based on the observation that the communication channel is shared by all members of a cluster. In a single-hop cluster, there can be only one transmission on the wireless channel at a given time. Therefore, the wireless channel can be modeled as a virtual node *CH* that executes one communication task at any time instance [6]. Hence, a single-hop cluster can be modeled as a star-network where all sensors only have connections with the virtual node *CH*. The communication latency between sensor nodes and *CH* is considered zero since all wireless communications are accounted for by the tasks executed on *CH*. Assuming that a cluster has p sensors that are denoted as $M = \{m_k\}$ $(0 \leq k < p)$, a cluster can be represented by a connected, undirected graph $G = (M', N)$, where the set $M' = M \cup \{CH\}$, and the set N denotes the links between the nodes of M'.

To implement the wireless channel model above, communication should be explicitly represented in task graphs. Thus, the DAG representation of applications is extended as follows: For a task v_i in a DAG, we replace the edges between v_i and its immediate successors with a *net* R_i. R_i represents delivery of $v_i's$ result to all of its immediate successors in the DAG. This extended DAG is a hypergraph and is referred to as *Hyper-DAG*. The example of converting the DAG in Fig. 1(a) to a Hyper-DAG is shown in Fig. 1(b). A Hyper-DAG is represented as $T' = (V', E')$, where $V' = \{\gamma_i\} = V \cup R$ denotes the new set of tasks to be scheduled and E' represents the dependencies between tasks with zero link cost. Here, $V = \{v_i\} = \{Computation\ Tasks\}$, and $R = \{R_i\} = \{Communication\ Task\}$.

Based on the network model and the Hyper-DAG representation of DAGs, HDMS has the following constraints when mapping the tasks of a Hyper-DAG: 1. Computation tasks can be assigned only on sensor nodes; 2. Communication tasks can have multiple copied assigned on sensor nodes and *CH*; 3. Communication tasks assigned on a sensor node have zero execution length and energy cost; 4. To meet *Dependency Constraint*, a non-entry computation task v_i assigned on a sensor m_k must have all of its immediate predecessors $pred(v_i)$, which are communication tasks, assigned on m_k.

With the constraints presented above, HDMS-based task mapping algorithms need to obey the following rules. If a computation task v_i does not have all of its predecessors on the target sensor m_k, the absent predecessor $v_n \in pred(v_i) - T(m_k)$ has to be duplicated to m_k: if $v_n \in T(CH)$, a copy of v_n is duplicated to m_k starting from $f_{n,CH}$; otherwise, a copy of v_n is first duplicated to CH from the source starting at the earliest available time of CH for v_n, then from CH to m_k. To schedule a copy of task v_i from sensor m_s to CH and from CH to sensor m_r are equivalent to scheduling data broadcast and data reception in the physical world, respectively. The sender m_s and receiver m_r of task v_i need to be updated with the corresponding energy consumption. This communication task duplication procedure is referred to as "Communication Scheduling".

4 Heuristics

In this section, E^2-CNPT, H-MinMin, and H-BottomsUp algorithms are introduced. These are implementations of CNPT [10], Min-Min, and Bottoms Up [9] algorithms using Hyper-DAGs following the constraints and the rules presented in Section 3.

E^2-CNPT: The E^2-CNPT algorithm is based on the Critical Nodes Parent Trees (CNPT) algorithm [10]. E^2-CNPT has two stages: *Listing Stage* and *Sensor Assignment Stage*. After the Listing Stage, all tasks in V' are enqueued into *mappable task queue* L in the order that the most critical path comes the first and a task is always enqueued after its predecessors. The listing procedure of E^2-CNPT is similar to that of CNPT except for the definitions of Earliest Start Time *EST* and Latest Start Time *LST*,

$$EST(v_i) = \max_{v_m \in pred(v_i)} \{EST(v_m) + w_m\}, \tag{1}$$

$$LST(v_i) = \min_{v_m \in succ(v_i)} \{LST(v_m)\} - w_i, \tag{2}$$

where w_i equals to the execution length on sensor nodes if $v_i \in V$ or to the execution length on *CH* if $v_i \in R$.

In the Sensor Assignment Stage, it is assumed that an increased number of sensors involved in computation decreases the schedule length. The sensor assignment algorithm iteratively searches the task mapping and scheduling solution with the maximum number of computing sensors under energy consumption constraints. Here, a computing sensor is a sensor that executes non-entry-tasks. The E^2-CNPT with q computing sensors is described as follows:

1. Dequeue a task v_i from the *mappable task queue* L created in *Listing Phase.*
2. Find the sensor m^o that gives the minimum Earliest Execution Start Time (EEST). A non-entry computation task $v_i \in V$ can only be assigned on one of the q computing sensors. If a task depends on its immediate predecessor assigned on another sensor, the "Communication Scheduling" procedure in Section 3 is executed to meet *Dependency Constraint.*

3. Assign v_i to m^o.
4. Repeat step 1 to 3 until L is empty.

Among the schedules with different number of computing sensors, the one with the minimum schedule length subject to the energy consumption constraint is chosen as the solution. If no schedule meets the energy consumption constraint, the solution with the minimum energy consumption is chosen.

H-MinMin: The Min-Min Algorithm in [9] is extended with a fitness function for task mapping and scheduling in WSNs. Let EB be the energy consumption constraint, $PE(v_i, m_k)$ be the partial energy consumption of the cluster after assigning task v_i to sensor m_k, f_{v_i,m_k} be the scheduled finish time of task v_i on sensor m_k, $MFT(v_i, m_k)$ be the maximum finish time of the tasks assigned prior to task v_i, $NPE(v_i, m_k) = PE(v_i, m_k)/EB$ be the normalized partial energy consumption of assigning v_i on m_k, and $NPT(v_i, m_k) = f_{v_i,m_k}/MFT(v_i, m_k)$ be the normalized partial execution time of assigning v_i on m_k. Using α as the weight parameter, the fitness function of assigning a task v_i on sensor m_k is defined as:

$$fitness(v_i, m_k) = \alpha \cdot NPE(v_i, m_k) + (1 - \alpha) \cdot NPT(v_i, m_k). \tag{3}$$

The H-MinMin algorithm is summarized as follows.

1. Initialize a mappable task list L with entry-tasks.
2. For each task $v_i \in L$, find the sensor m_i^o that gives the minimum fitness value $fitness(v_i, m_i^o)$. The "Communication Scheduling" procedure is executed to meet *Dependency Constraint* if necessary and the corresponding energy consumption is counted into the $fitness(v_i, m_k)$ calculation.
3. From the task/sensor pairs found in step 2, find the pair (v^o, m^o) with the minimum fitness value.
4. Remove task v^o from L and assign v^o to sensor m^o.
5. Update L with any new mappable tasks after step 4.
6. Repeat steps 2 to 5 until all tasks are scheduled.

In the algorithm above, a "mappable task" is either an entry-task or a task that has all *immediate predecessors* already assigned. Among the schedules with different values of α ($\alpha = 0.1 \cdot i, i = 0, 1, ...10$), the schedule with the minimum schedule length subject to the energy consumption constraint is chosen as the solution. If no schedule meets the energy consumption constraint, the one with the minimum energy consumption is chosen.

H-BottomsUp: The fitness function of the modified Bottoms Up [9] is the same as that of the H-MinMin algorithm. Different from H-MinMin, a mappable task in Bottoms Up is either an exit-task or a task that has all *immediate successors* already mapped. The H-BottomsUp algorithm can be outlined as following:

1. Assign levels to the tasks: entry-tasks have level zero; a non-entry-task is one level higher than the maximum of its immediate predecessors' levels.
2. From the highest level to the lowest level, consider all mappable tasks within each individual level.

3. For each mappable task v_i in the current level, find the sensor m_k that gives the minimum $fitness(v_i, m_k)$. Execute the "Communication Scheduling" procedure to meet *Dependency Constraint* if necessary.
4. From the task/sensor pairs found in step 3, find the pair (v^o, m^o) with the minimum fitness value, assign v^o to m^o.
5. Repeat step 3 to 4 until all tasks in the present level are assigned.
6. Repeat step 2 to 5 until all tasks are mapped.
7. Schedule the tasks in the reverse order that they were assigned.

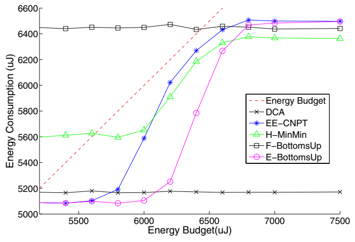
In the algorithm above, the "Communication Scheduling" procedure is similar to that presented in Section 3 except that the absent *immediate successors* of a computation task are duplicated instead of the absent immediate predecessors. In [9], a weight factor $\alpha = 0.52$ is reported to provide best performance under all scenarios. In this paper, the H-BottomsUp algorithm with $\alpha = 0.52$ is evaluated (referred to as F-BottomsUp). To further study the performance of the H-BottomsUp algorithm in WSNs, the H-BottomsUp algorithm is also executed with α ranging from 0 to 1 in steps of 0.1 (referred to as E-BottomsUp).

5 Performance Evaluation

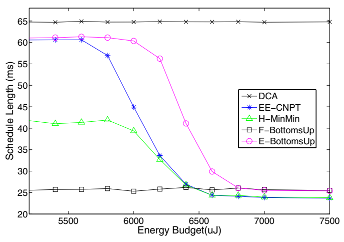
The performance of E^2-CNPT, Min-Min, F-BottomsUp, and E-BottomsUp algorithms are evaluated through simulations. The performance of the distributed computation architecture (referred to as DCA) [1] [2] is also evaluated as a benchmark. DCA is extended such that several sensors perform entry-tasks and send intermediate results to cluster heads for further processing. We have run simulations to investigate the following aspects: 1. Effect of energy consumption constraints; 2. Effect of number of tasks in applications; 3. Effect of communication load between tasks; 4. Comparison of Heuristic Execution Times.

In these simulations, we observe energy consumption and schedule lengths. The energy consumption includes computation and communication energy expenditure of all sensors. The schedule length is defined as the finish time of the exit-task of an application. In the last set of experiments, the execution times of E^2-CNPT, H-MinMin, F-BottomsUp, and E-BottomsUp are normalized by the execution time of DCA, which serves as a relative metric of the algorithm complexity. The simulation results presented in this section correspond to the average of two hundred independent runs.

Simulation Parameters and Energy Consumption Model: In our simulation study, the bandwidth of the channel is set to 1Mb/s and the transmission range to 10 meters. The sensors have the clock frequency of 100 MHz. The energy consumptions of transmitting and receiving l-bit data over distance d are defined as $E_{tx}(l, d) = E_{elec} \cdot l + \varepsilon_{amp} \cdot l \cdot d^2$ and $E_{rx}(l) = E_{elec} \cdot l$, where $E_{elec} = 50$ nJ/b and $\varepsilon_{amp} = 10$ pJ/b/m^2 [11]. The energy consumption of executing N clock cycles (CC) with CPU frequency f is given as: $E_{comp}(V_{dd}, f) = NCV_{dd}^2 + V_{dd}(I_o e^{\frac{V_{dd}}{nV_T}})(\frac{N}{f})$ and $f \simeq K(V_{dd} - c)$, where V_T is the thermal voltage and C, I_o, n, K and c are processor dependent parameters [2]. In our simulation,



(a) Energy Consumption



(b) Schedule Length

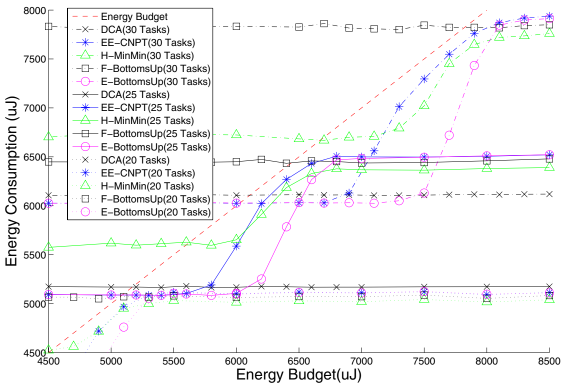
Fig. 2. Effect of Energy Budget

we consider the StrongARM SA-1100 microprocessor, where $V_T = 26$ mV, $C = 0.67$ nF, $I_o = 1.196$ mA, $n = 21.26$, $K = 239.28$ MHz/V and $c = 0.5$ V [1], [2].

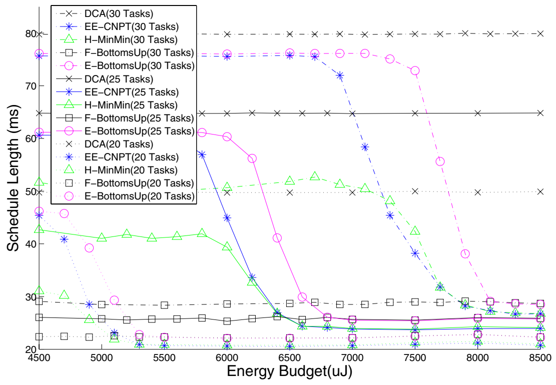
Simulations are run on randomly generated DAGs, which are created based on three parameters, namely, the number of tasks *numTask*, the number of entry-tasks *numEntry*, and the maximum number of immediate predecessors *maxPred*. The number of each non-entry task's immediate predecessors, the computation load, and the resulting data volume of a task are uniformly distributed over [1, *maxPred*], [300 KCC ±10%], and [800 bits ±10%], respectively.

Effect of Energy Consumption Constraints: We investigate the effect of energy consumption constraints with randomly generated DAGs. The parameters of DAGs considered for this set of simulations are $numTask = 25$, $numEntry = 6$, and $maxPred = 3$. The energy consumption and schedule length are observed for different available energy levels (referred to as Energy Budget). According to Fig. 2, when the energy budget is small, H-MinMin and F-BottomsUp fail to meet energy constraints while E^2-CNPT and E-BottomsUp succeed. When the energy budget increases, the schedule lengths of E^2-CNPT, H-MinMin, and E-BottomsUp decrease while DCA and F-BottomsUp hardly improve their schedule lengths. With sufficient energy supply, E^2-CNPT and H-MinMin improve schedule lengths up to 63% while F-BottomsUp and E-BottomsUp obtain 60% improvement in comparison to DCA. From the simulation results, we can see that E^2-CNPT comprehensively outperforms H-MinMin, F-BottomsUp and E-BottomsUp with respect to meeting energy constraints and shortening schedule lengths.

Effect of Number of Tasks in Applications: To investigate the effect of number of tasks in applications, three sets of simulations are run on randomly generated DAGs with 20, 25 and 30 tasks ($numEntry = 6$, $maxPred = 3$). According to the simulation results in Fig. 3, energy consumption and schedule length are dominated by the number of tasks. When the number of tasks increases, the energy consumption of DCA and F-BottomsUp increases proportionally. E^2-CNPT, H-MinMin, and E-BottomsUp on the other hand adapt themselves to the increasing energy budget. For small energy budgets, the schedule lengths of all five algorithms increase when the number of tasks increases. However, for



(a) Energy Consumption

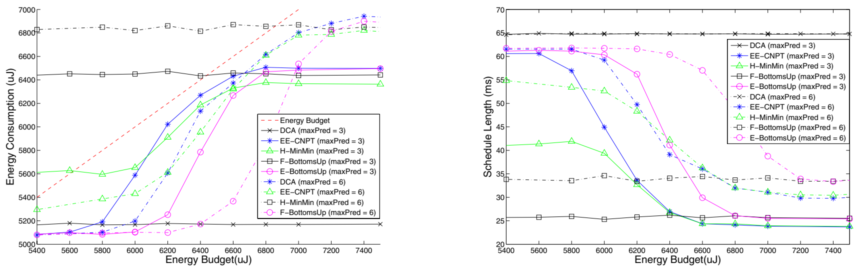


(b) Schedule Length

Fig. 3. Effect of Number of Tasks (30 tasks VS 25 tasks VS 20 tasks)

E^2-CNPT, H-MinMin, and E-BottomsUp, the schedule lengths reduce when the energy budget increases. However, the schedule lengths of DCA and F-BottomsUp increase proportionally when the number of tasks increases.

Effect of Inter-task Dependency: Inter-task dependency is determined by the in/out degree of application DAGs. Two sets of simulations with $maxPred = 3$ and $maxPred = 6$ ($numTask = 25$, $numEntry = 6$) are executed. According to Fig. 4, inter-task dependency has almost no effect over the performance of DCA. Both of the energy consumption and schedule length increase when inter-task dependency increases in E^2-CNPT, H-MinMin, F-BottomsUp, and E-BottomsUp. The robustness of DCS against inter-task dependency change is because DCA has most tasks executed on cluster heads with the least need for communication. Regarding E^2-CNPT, H-MinMin, F-BottomsUp, and E-BottomsUp, increasing the in/out degree of DAGs does not introduce new communication tasks but



(a) Energy Consumption (b) Schedule Length

Fig. 4. Effect of Inter-task Dependency (25 tasks)

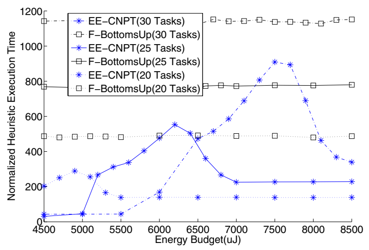


Fig. 5. Normalized Heuristic Execution Time(30 tasks VS 25 tasks VS 20 tasks)

increases the dependency between a communication task and its immediate successors. Greater dependency degree between tasks leads to a higher number of communication tasks scheduled on CH and less parallelism between sensors, which causes more energy consumption and longer schedules.

Comparison of Heuristic Execution Times: The heuristic execution times are investigated with three sets of simulations run on randomly DAGs with 20, 25 and 30 tasks ($numEntry = 6$, $maxPred = 3$). For fair comparison, the execution times of E^2-CNPT, H-MinMin, F-BottomsUp, and E-BottomsUp are normalized by the average execution time of DCA with 20 tasks. According to the simulation, the normalized execution times of H-MinMin and E-BottomsUp are over 5000 and significantly higher than E^2-CNPT and F-BottomsUp. Thus, only the normalized execution times of E^2-CNPT and F-BottomsUp are shown in Fig. 5. The execution time of F-BottomsUp almost keeps constant with different energy budget. When the energy budget increases, the execution time of E^2-CNPT first increases, then decreases after reaching a peak value. E^2-CNPT iteratively searches computing sensor space for optimal solutions. The searching algorithm converges faster in the scenarios with limited or sufficiently large energy budget and converges slower with intermediate scenarios. When the number of tasks increases, the execution time of F-BottomsUp proportionally increases while E^2-CNPT adapts according to energy budgets. In all scenarios, the execution time of E^2-CNPT is shorter than that of F-BottomsUp with the same number of tasks. Thus, E^2-CNPT has better scalability compared with H-Min-Min, F-BottomsUp, and E-BottomsUp.

6 Conclusion

In this paper, a wireless network model, the Hyper-DAG application representation, and communication scheduling rules are introduced for task mapping and scheduling in wireless sensor networks. Three HDMS algorithms are presented, which aim to minimize schedule lengths of applications under energy consumption constraints. Simulations with randomly generated DAGs show that E^2-CNPT provides superior performance in comparison with DCA, H-MinMin, F-BottomsUp, and E-BottomsUp algorithms.

References

1. Shih, E., Cho, S., Ickes, N., Min, R., Sinha, A., Wang, A., Chandrakasan, A.: Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In: Proc. of ACM MobiCom'01. (2001) 272–286
2. Wang, A., Chandrakasan, A.: Energy-efficient DSPs for wireless sensor networks. IEEE Signal Processing Magazine (2002) 68–78
3. Kumar, R., Tsiatsis, V., Srivastava, M.B.: Computation hierarchy for in-network processing. In: Proc. of the 2nd ACM international conference on Wireless Sensor Networks and Applications (WSNA'03). (2003) 68–77
4. Dogan, A., Özgüner, F.: Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogenous computing. IEEE Transaction on Parallel and Distributed Systems **13** (2002) 308–323
5. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co. (1979)
6. Giannecchini, S., Caccamo, M., Shih, C.S.: Collaborative resource allocation in wireless sensor networks. In: Proc. of Euromicro Conference on Real-Time Systems (ECRTS'04). (2004) 35–44
7. Basu, P., Ke, W., Little, T.D.C.: Dynamic task-based anycasting in mobile ad hoc networks. Mobile Networks and Applications **8** (2003) 593–612
8. Kumar, R., Wolenetz, M., Agarwalla, B., Shin, J., Hutto, P., Paul, A., Ramachandran, U.: DFuse: A framework for distributed data fusion. In: Proc. of The ACM Conference on Embedded Networked Sensor Systems (SenSys'03). (2003) 114–125
9. Shivle, S., Castain, R., Siegel, H.J., Maciejewski, A.A., Banka, T., Chindam, K., Dussinger, S., Pichumani, P., Satyasekaan, P., Saylor, W., D.Sendek, Sousa, J., Sridharan, J., Sugavanam, P., Velazco, J.: Static mapping of subtasks in a heterogeneous ad hoc grid environment. In: Proc. of Parallel and Distributed Processing Symposium. (2004)
10. Hagras, T., Janecek, J.: A high performance, low complexity algorithm for compile-time job scheduling in homogeneous computing environments. In: Proc. of International Conference on Parallel Processing Workshops (ICPPW'03). (2003) 149–155
11. Heinzelman, W.B., Chandrakasan, A., Balakrishnan, H.: An application-specific protocol architecture for wireless microsensor networks. IEEE Transactions on Wireless Communications **1** (2002) 660–670