Real-Time Task Mapping and Scheduling for Collaborative In-Network Processing in DVS-Enabled Wireless Sensor Networks

Yuan Tian, Jarupan Boangoat, Eylem Ekici, and Füsun Özgüner
Department of Electrical and Computer Engineering, The Ohio State University
Email: {tiany,bea,ekici,ozguner}@ece.osu.edu

Abstract

With the increasing importance of energy consumption considerations and new requirements of emerging applications, in-network processing of information gains recognition as a viable solution for Wireless Sensor Networks (WSNs). The required processing capability can be achieved through locally collaborative information processing among sensors. Task mapping and scheduling plays an important role in efficient collaborative information processing. Although task mapping and scheduling in wired networks of processors has been well studied in the past, its counterpart for WSNs remains largely unexplored. In this paper, a task mapping and scheduling solution for real-time applications in WSNs, Real-time Task Mapping and Scheduling (RT-MapS), is presented. RT-MapS incorporates wireless channel modeling, Hyper-DAG extension, concurrent task mapping, communication and computation scheduling, and Dynamic Voltage Scaling (DVS) methods. Simulation results show significant performance improvements compared with existing mechanisms in terms of providing deadline guarantee with minimum energy consumption.

I. Introduction

In-network processing has been recognized as a viable approach to significantly decrease network energy consumption [1] [2]. In-network processing is also required by many emerging applications with considerable computation demands in resource-constrained WSNs, such as video sensor networks [3] composed of wireless sensors equipped with cameras. Since multimedia processing generally involves computationally intensive operations with real-time requirements, the resulting need for computation power creates new challenges for WSN design. To provide the demanded computation power for applications, which is generally more than the capacity of a single sensor, a promising solution is to have sensors collaboratively process information. *Task mapping and scheduling* plays an essential role in parallel processing by solving the following problems subject to application requirements:

- Assignment of tasks to sensors;
- Execution sequence of tasks on sensors;
- Communication schedule between sensors.

Existing task mapping and scheduling solutions for wired networks cannot be implemented directly in WSNs as the generally assumed point-to-point connections between all nodes do not exist in WSNs. Furthermore, many solutions do not explicitly consider energy consumption during communication and task execution, which is one of the major constraints in WSNs. Thus, task mapping and scheduling remains largely unexplored for WSNs.

Depending on applications and network scale, task mapping and scheduling can be achieved either network-wide or in a localized manner in WSNs. In large-scale WSNs, global optimization of task mapping and scheduling is a costly task. Furthermore, events of interest in such networks generally occur in remote regions that only local sensors can detect. Thus, local information processing, and consequently, *localized task mapping and scheduling*, is more suitable for large-scale WSNs. In localized task mapping and scheduling, solutions focus on performance optimization applied to clusters alone.

In [4], an online task scheduling mechanism (CoRAl) is proposed to allocate network resources between the tasks of periodic applications in WSN clusters iteratively: The frequencies of the tasks on each sensor are optimized subject to the previously evaluated upper-bound execution frequencies. However, CoRAl does not address mapping tasks to sensor nodes. Distributed Computing Architecture (DCA) is proposed in [1], which executes low level tasks on sensing sensors and offload all other high level processing tasks to cluster heads. However, processing high level tasks can still exceed the capacity of cluster heads' com-

1-4244-0054-6/06/$20.00

putation power. Furthermore, application-specific design of these solutions limit their implementation for generic applications.

Task mapping and task scheduling have been jointly considered for mobile computing [5] and for WSNs [6] [7] recently. Task mapping and scheduling heuristics are presented in [5] for heterogeneous mobile ad hoc grid environments. However, the communication model adopted in [5] is not well-suited for WSNs, which assumes individual channels for each node, and concurrent data transmission and reception capacity of every node. The EcoMapS algorithm in [6] aims to minimize the schedule length subject to the energy consumption constraint in single-hop clustered WSNs. However, EcoMapS does not provide execution deadline guarantees for applications. In [7], Energy-balanced Task Allocation (EbTA) is introduced to minimize balanced energy consumption subject to application deadline constraints. In [7], communications over multiple wireless channels are modeled as additional linear constraints of an Integer Linear Programming (ILP) problem, and a heuristic algorithm with Dynamic Voltage Scaling (DVS) mechanism is presented. However, the communication scheduling model in [7] does not exploit the broadcast nature of wireless communication, which can conserve energy and time expenditure.

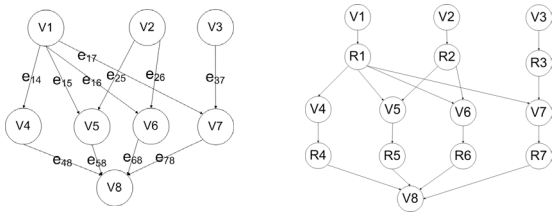
In this paper, we propose localized cross-layer **R**eal-time **T**ask **Map**ping and **S**cheduling solutions (RT-MapS) for DVS-enabled WSNs. We consider deadline-constrained applications executed in a single-hop cluster of a homogeneous WSN. The design objective of RT-MapS is to *minimize energy consumption subject to application deadline constraints*. In RT-MapS, communication and computation are jointly scheduled in two phases: *Task Mapping and Scheduling Phase* and *DVS Phase*. In the *Task Mapping and Scheduling Phase*, two low-complexity task mapping and scheduling algorithms, CNPT [8] and Min-Min algorithm [9] [5], are extended and implemented. The extended CNPT and Min-Min algorithms incorporate our proposed communication scheduling algorithm with the objective of minimizing energy consumption subject to deadline constraints, and broadcasting capability is exploited to conserve energy consumption. The Dynamic Voltage Scaling (DVS) technique is implemented in the *DVS phase* to further reduce energy consumptions.

II. Preliminaries

A. Network Assumptions

Our proposed task mapping and scheduling mechanism is designed for applications executed within a single-hop cluster of wireless sensor networks. The following assumptions are made for the wireless sensor networks:

- Each cluster executes an application which is either assigned during the network setup time or remotely distributed during the network operation.
- Once assigned, applications are independently executed within each cluster as many times according to the application requirements. With applications arrivals, cluster heads create the schedules for application communication and computation within clusters.
- Intra-cluster communication is isolated from other clusters through time division or channel hopping.
- Sensors are equipped with DVS processors with finite number of CPU speeds and supply voltage levels. The overhead of speed and voltage adjustment is assumed to be negligible.



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

Fig. 1. DAG and Hyper-DAG Examples

It should be noted that while the intra-cluster communication is isolated from each other, the communication across clusters is assumed to be handled over common time slots or channels orthogonal to those used inside a cluster. As such, information flow across the network is not hindered by intra-cluster communication isolation.

B. Application and Energy Consumption Model

To have an application-independent solution, we represent applications executed in clusters with Directed Acyclic Graphs (DAG). A DAG *T = (V,E)* consists of a set of vertices *V* representing the tasks to be executed and a set of directed edges *E* representing dependencies among tasks. The edge set E contains directed edges e_{ij} for each task $v_i \in V$ that task $v_j \in V$ depends on. For an edge e_{ij}, v_i is called the immediate predecessor of v_j, and v_j is called the immediate successor of v_i. A task without immediate predecessors is called an *entry-task* and a task without immediate successors is called an *exit-task*. Fig. 1(a) shows an example of a DAG.

If a task v_j scheduled on one node depends on a task v_i scheduled on another node, a communication between these nodes is required. In such a case, v_j cannot start its execution until the communication is completed and the result of v_i is received. However, if both tasks are assigned on same node, the result delivery latency is considered to be zero and v_j can start to execute after v_i is finished.

This execution dependency between tasks is referred to as *Dependency Constraint* throughout the paper.

The energy consumption of transmitting and receiving l-bit data over a distance d that is less than a threshold d_o are defined as $E_{tx}(l,d)$ and $E_{rx}(l)$, respectively:

$$E_{tx}(l,d) = E_{elec} \cdot l + \varepsilon_{amp} \cdot l \cdot d^2, \quad \text{(II-B.1)}$$

$$E_{rx}(l) = E_{elec} \cdot l, \quad \text{(II-B.2)}$$

where E_{elec} and ε_{amp} are hardware parameters [1] [10].

The energy consumption of executing N clock cycles at CPU frequency f and supply voltage V_{dd} is given as:

$$E_{comp}(V_{dd}, f) = NCV_{dd}^2 + V_{dd}(I_o e^{\frac{V_{dd}}{nV_T}})(\frac{N}{f}), \quad \text{(II-B.3)}$$

$$f \simeq K(V_{dd} - c), \quad \text{(II-B.4)}$$

where V_T is the thermal voltage and C, I_o, n, K and c are processor dependent parameters [11] [1].

C. Problem Statement

The task mapping and scheduling problem is to find a set of task assignments and their execution sequences on a network that minimizes an objective function. Let $H^x = \{h_1^x, h_2^x, ..., h_n^x\}$ denote a task mapping and scheduling solution of the application DAG *T* on a network *G*, where x is the index of the task mapping and scheduling solution space. Each element $h_i^x \in H^x$ is a tuple of the form $(v_i, m_k, s_{i,m_k}, t_{i,m_k}, f_{i,m_k}, c_{i,m_k})$, where m_k represents the node to which task v_i is assigned, s_{i,m_k} and f_{i,m_k} represent the start time and finish time of v_i, and t_{i,m_k} and c_{i,m_k} represent the execution length and energy consumption of v_i on node m_k, respectively. The design objective of this paper is to find an $H^o \in \{H^x\}$ that has the minimum energy consumption under the deadline constraint, which can be formulated as follows:

$$\min energy(H^o) = \sum_{i,k} c_{i,m_k}; \quad \text{(II-C.5)}$$

$$\text{subject to } length(H^o) = \max_{i,k} f_{i,m_k} \leq DL, \quad \text{(II-C.6)}$$

where $length(H)$ and $energy(H)$ are the schedule length and energy consumption of H, respectively, and DL is the deadline of the application. DAG scheduling problem is shown to be an NP-complete problem in general [12]. Therefore, heuristic algorithms are needed to solve this problem in polynomial time.

Some notations are listed here for convenience:

- $pred(v_i)$ and $succ(v_i)$ denote the immediate predecessors and immediate successors of task v_i respectively
- $m(v_i)$ denotes the node on which v_i is assigned
- $T(m_k)$ denotes the tasks assigned on node m_k
- $T_{st}^{ft}(m_k)$ denotes the tasks assigned on node m_k during the time interval $[st, ft]$

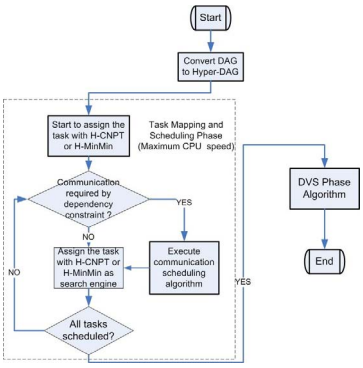


Fig. 2. Flowchart of RT-MapS

III. The Proposed RT-MapS Solution

The proposed RT-MapS sloution is demonstrated with the flowchart in Fig. 2. RT-MapS has two phases: *Task Mapping and Scheduling Phase* and *DVS Phase*. In the *Task Mapping and Scheduling Phase*, communication and computation tasks are scheduled. Two low-complexity task mapping and scheduling algorithms, CNPT [8] and Min-Min [9] [5] are extended and implemented with the objective of minimizing the energy consumption subject to the deadline constraint. A *communication scheduling algorithm* is developed based on our wireless channel model and Hyper-DAG representation of applications. The proposed communication scheduling algorithm is then embedded in the execution of the extended CNPT and Min-Min algorithms to satisfy the *Dependency Constraint*. The energy consumption is then further reduced in the *DVS Phase*. In the following sections, the main components of the RT-MapS solution, namely, wireless channel modeling and Hyper-DAG extension, communication scheduling algorithm, Hyper-DAG based CNPT and Min-Min algorithms (referred to as H-CNPT and H-MinMin), and DVS algorithm, are presented.

A. Wireless Channel Modeling and Hyper-DAG Extension

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 $\mathcal{C}$ that executes one communication task at any time instance. Similar to [4], a cluster can be modeled as a star-network where all sensors only have connections with the virtual node $\mathcal{C}$. The communication latency between sensor nodes and $\mathcal{C}$ can be considered zero since all wireless communications are accounted for by the tasks executed

on $\mathcal{C}$. Assuming that the cluster has p sensors 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 $M' = M \cup \{\mathcal{C}\}$, and the set *N* denotes the links between the nodes of M'. With the virtual node representation of $\mathcal{C}$, communication contention can be effectively avoided by serially scheduling communications on $\mathcal{C}$.

To implement this channel model, communication events between computation tasks should be explicitly represented in task graphs. To accomplish this, 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. The weight of R_i equals to the resulting data volume of v_i. R_i represents the communication task to send the result of v_i to its immediate successors in the DAG. This extended DAG is a hypergraph and is referred to as *Hpyer-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. Here, $V = \{v_i\}$ = {*Computation Tasks*}, and $R = \{R_i\}$ = {*Communication Tasks*}.

With Hyper-DAGs, the *Dependency Constraint* in Section II-B is rephrased as follows: If a computation task v_j scheduled on node m_k depends on a communication task v_i scheduled on another node, a copy of the communication task v_i needs to be scheduled to m_k, and v_j cannot start to execute until all of its immediate predecessors are received on the same node.

With the joint assistance of the channel model and the Hyper-DAG representation, exclusive channel access constraints and broadcast delivery of results are incorporated into task dependency in a compact way. The broadcast nature of the wireless channel can also be leveraged to relay information generated by a task to all its immediate successors in a single transmission rather than multiple, sequential transmissions. This approach both reduces the execution time as well as the energy consumption.

B. Communication Scheduling Algorithm

To meet the *Dependency Constraint* in Hyper-DAG scheduling, communication between nodes is required if a computation task depends on a communication task assigned on another node. Our communication scheduling algorithm is presented in this section. As we shall see in Section III-C, the communication scheduling algorithm is integrated into the execution of our Hyper-DAG task mapping and scheduling algorithms, H-CNPT and H-MinMin.

Based on the Hyper-DAG and the channel model presented in Section III-A, scheduling communication between single-hop neighbors is equivalent to first duplicating a communication task from the sender to $\mathcal{C}$, and then from $\mathcal{C}$ to the receiver. If the requested communication task has been scheduled from the sender to another node before, the receiver will directly duplicate the communication task from $\mathcal{C}$. This process is equivalent to receiving broadcast data, which can lead to significant energy saving compared with multiple unicast. The detailed description of the communication scheduling algorithm is presented below.

Input: Communication task v_i, sender m_s, and receiver m_r
Output: Schedule of duplicating v_i from m_s to m_r
CommTaskSchedule(v_i,m_s,m_r):
1. Find a copy of v_i: $v_i^c \in T(\mathcal{C})$
2. **IF** v_i^c does not exist
3. Find $v_i \in T(m_s)$, and time interval [st,ft]:
4. $T_{st}^{ft}(\mathcal{C}) = \emptyset$, $ft - st \geq t_{v_i,\mathcal{C}}$
5. $st \geq f_{v_i,m_s}$, $st = \min$
6. Schedule a copy of v_i to $\mathcal{C}$:/*data transmission*/
7. $v_i^c \in T(\mathcal{C})$, $s_{v_i^c,\mathcal{C}} \leftarrow st$
8. Update the energy consumption of m_s
9. Schedule a copy of v_i^c to m_r: /*data reception*/
10. $v_i^k \in T(m_k)$, $s_{v_i^k,m_k} \leftarrow f_{v_i^c,\mathcal{C}}$
11. Update the energy consumption of m_r
12.**ELSE**
13. Schedule a copy of v_i^c to m_r:/*data reception*/
14. $v_i^k \in T(m_k)$, $s_{v_i^k,m_k} \leftarrow f_{v_i^c,\mathcal{C}}$
15. Update the energy consumption of m_r

In the algorithm above, Steps 2-11 stands for originating a new communication from m_s to m_r, and Steps 12-15 represents reception of a broadcast data. Compared with originating a new communication, the broadcast reception method leads to energy saving of one data transmission for each additional data reception. Note that copying a communication task from $\mathcal{C}$ to a node incurs energy consumption associated with reception, but does not introduce additional time consumption.

C. Task Mapping and Scheduling with H-CNPT and H-MinMin Algorithm

In the *Task Mapping and Scheduling Phase* of RT-MapS, the tasks of Hyper-DAGs are mapped and scheduled on sensors. During task mapping, several constraints have to be satisfied. These constraints together with the *Dependency Constraint* are represented as follows.

- Computation tasks can be assigned only on sensor nodes, ie., $\forall \gamma_i \in V : t_{i,\mathcal{C}} = \infty, c_{i,\mathcal{C}} = \infty$
- Communication tasks can be assigned both on sensors and $\mathcal{C}$
- If $v_i \in V$ and $pred(v_i) \neq \emptyset$, then $pred(v_i) \subset T(m(v_i))$ and $s_{v_i,m(v_i)} \geq \max f_{pred(v_i),m(v_i)}$

To meet the *Dependency Constraint* during task mapping and scheduling, if a computation task depends on a

communication task assigned on another sensor node, the *communication scheduling algorithm* will be executed to duplicate the absent communication task. With the Communication Scheduling Algorithm and the task mapping constraints presented above, task mapping and scheduling in single-hop wireless networks can be tackled as a generic task mapping and scheduling problem with additional constraints. This problem is NP-complete in general [12] and heuristic algorithms are needed. In this section, two task mapping and scheduling algorithms, H-CNPT algorithm and H-MinMin algorithm are presented with the objective of minimizing energy consumption subject to deadline constraints. To guarantee deadlines, sensors are scheduled with the maximum CPU speed f_{cpu}^{max}.

Before presenting the H-CNPT and H-MinMin algorithms, we first introduce a concept of *computing sensor*: A computing sensor is a sensor that can execute non-entry tasks as well as entry-tasks. The concept of computing sensor is an intuitive extension of DCA in [1], where only one sensor, ie. the cluster head, in a cluster can execute high level tasks. In RT-MapS, there can be more than one computing sensors to speed up execution. However, this approach generally consumes more energy because of the increased volume of communication between sensors. Thus, the number of computing sensors shall only be increased to the extend of satisfying deadline constraints. In RT-MapS, H-CNPT and H-MinMin will iteratively search the optimal schedule with different number of computing sensors subject to deadline constraints.

1) H-CNPT Algorithm: The strategy of H-CNPT is to assign the tasks along the most critical path first to the nodes with earliest execution start times. By adjusting the number of computing sensors in each scheduling iteration and choosing the schedule with the minimum energy consumption under the deadline constraint, the design objective of H-CNPT is achieved. Similar to CNPT, H-CNPT also has two stages: *listing stage* and *sensor assignment stage*. In the *listing stage*, tasks are sequentialized into a queue L such that the most critical path comes the first and a task is always enqueued after its immediate predecessors. In the *sensor assignment stage*, the tasks will be dequeued from L and assigned to the sensors with the minimum execution start time. Several scheduling iterations will be run in the sensor assignment stage with different number of computing sensors, and only the optimal schedule is chosen. The *listing stage* and *sensor assignment stage* of H-CNPT are introduced individually as follows.

Listing Stage: The Listing Stage of H-CNPT is similar to that of CNPT [8] except that there are two types of tasks in H-CNPT: Computation task and communication task. Thus, the formulas to calculate the Earliest Start Time $EST(v_i)$ and the Latest Start Time $LST(v_i)$ of task v_i are different from those of CNPT, and are presented as follows:

$$EST(v_i) = \max_{v_m \in pred(v_i)} \{EST(v_m) + t_m\}, \quad \text{(III-C.7)}$$

$$LST(v_i) = \min_{v_m \in succ(v_i)} \{LST(v_m)\} - t_i, \quad \text{(III-C.8)}$$

where t_i equals to the execution length on sensor nodes if $v_i \in V$ or to the execution length on $\mathcal{C}$ if $v_i \in R$. After the Listing Phase, the task graph is sequentialized into L and is ready for the Sensor Assignment Phase. The details of the Listing Stage can be found in [8].

Sensor Assignment Stage: In the Sensor Assignment Stage, H-CNPT will iteratively search the schedule space with different number of computing sensors. Among these schedules, the one with the minimum energy consumption under the deadline constraint is chosen as the solution. If no schedule meets the deadline constraint, the schedule with the minimum schedule length is chosen. The detailed description of the H-CNPT algorithm is given below.

Input: Task queue L; number of available sensors in the cluster p; deadline DL

Output: Schedule H^o of tasks in L with minimum schedule length under energy budget constraint

H-CNPT Algorithm:

1. $L_{min} \leftarrow \infty$ /*minimum schedule length*/
2. $E^o \leftarrow \infty$ /*optimal energy consumption*/
3. **FOR** q = 1 to p
4. H = SingleCNPT(L,q)
6. **IF** $length(H) < L_{min}$
7. $L_{min} \leftarrow length(H)$; $H_{min} \leftarrow H$
8. **IF** $length(H) \leq DL$ and $energy(H) < E^o$
9. $E^o \leftarrow energy(H)$; $H^o \leftarrow H$
10. **IF** $L_{min} \leq DL$
11. Return H^o
12. **ELSE**
13. Return H_{min}

In the H-CNPT algorithm above, SingleCNPT(L,q) is a single round of task scheduling that schedules the tasks in L with q computing sensors, where q is the total number of available computing sensors. The actual number of computing sensors in use can be smaller than q depending on the application and the scheduling algorithm. The core of SingleCNPT(L,q) is the extended CNPT *processor assignment algorithm*. The basic strategy of the algorithm is to assign tasks to the sensor with the minimum Earliest Execution Start Time (EEST). During task scheduling, *Dependency Constraint* must be satisfied via communication scheduling. SingleCNPT(L,q) is described as follows.

Input: Task queue L; number of computing sensors q

Output: Schedule H of tasks in L

SingleCNPT:

while L is not empty

1. Dequeue v_i from L
2. **IF** $v_i \in R$ /* communication task */
3. Assign v_i to node $m(pred(v_i))$

4. **ELSE IF** $pred(v_i) = \emptyset$ /*entry-tasks*/
5. Assign v_i to node m_i^o with min $EAT(m_i^o)$
6. **ELSE** /* non-entry computation tasks*/
7. **FOR** computing sensors $\{m_k\}$
8. Calculate EEST(v_i, m_k) with a copy of current schedule:
9. **IF** $pred(v_i) \subseteq T(m_k)$
10. EEST(v_i, m_k) $\leftarrow \max(EAT(m_k), f_{pred(v_i),m_k})$
11. **ELSE** /*communication between sensors is needed*/
12. **FOR** $v_n \in pred(v_i) - T(m_k)$
13. CommTaskSchedule(v_n,$m(v_n)$,m_k)
14. EEST(v_i, m_k) $\leftarrow \max(EAT(m_k), f_{pred(v_i),m_k})$
15. Keep the schedule with minimum EEST(v_i, m^o)
16. Schedule v_i on m^o: $s_{v_i,m^o} \leftarrow EEST(v_i, m^o)$

In the algorithm above, $EAT(m_k)$ is the Earliest Available Time of node m_k, and EEST(v_i, m_k) is the Earliest Execution Start Time of v_i on sensor m_k. Different from EST, EEST represents the actual execution start time of a task if assigned on a sensor node.

2) H-MinMin Algorithm: Similar to H-CNPT, H-MinMin also searches for a schedule with optimal number of computing sensors that has the smallest energy consumption subject to the deadline constraint. The H-MinMin's optimal number of computing sensors searching algorithm is the same as the **H-CNPT Algorithm** in Section III-C.1 except that the input of the H-MinMin algorithm is the Hyper-DAG instead of the task queue L, and the core of the searching algorithm is the **SingleMinMin** instead of the **SingleCNPT**. In the following, we introduce the procedure SingleMinMin(Hyper-DAG,q) that schedules the tasks of the Hyper-DAG with q computing sensors.

The core of the SingleMinMin algorithm is the fitness function. For each task-node combination (v,m), the fitness function $fit(m, k, \alpha)$ indicates the combined cost in time and energy domain of assigning task v to node m, where α is the weight parameter trading off the time cost for the energy consumption cost. At each step of the SingleMinMin algorithm, the task-node combination that gives the minimum fitness value among all combinations is always assigned first. To extend and describe the fitness function of the Min-Min Algorithm in [5], the following notations are introduced first:

- $f_{v,m}$ is the scheduled finish time of v on m
- $PEA(v, m)$ is the amount of application energy consumption before assigning v
- $PE(v, m)$ is the energy consumption after assigning v on m, which includes the computation energy consumption and communication energy consumption
- $NPT(v, m)$ is the normalized partial execution time of assigning v on m: $NPT(v, m) = f_{v,m}/DL$
- $NPE(v, m)$ is the normalized energy consumption of assigning v on m: $NPE(v, m) = PEA(v, m)/PE(v, m)$

Thus, the fitness of assigning v on m with α is defined as:

$$fit(v, m, \alpha) = \alpha \cdot NPT(v, m) + (1 - \alpha) \cdot NPE(v, m). \quad \text{(III-C.9)}$$

The SingleMinMin Algorithm is presented below. In the description of SingleMinMin, a "mappable" task is either an entry-task or a task that has all immediate predecessors already been assigned, and the "mappable task list" is the list that contains currently mappable tasks of the Hyper-DAG.

Input: Hpyer-DAG; number of computing sensors: q
Output: Schedule H of tasks in Hyper-DAG
SingleMinMin Algorithm:
1. **FOR** $\alpha = 0$; $\alpha \leq 1.0$; $\alpha += 0.1$
2. **FOR** entry-tasks v_i
3. Assign v_i on node m_i^o with min $EAT(m_i^o)$
4. Assign $succ(v_i)$ on m_i^o
5. Initialize the mappable task list L
6. **WHILE** L is not empty, with a copy of current schedule:
7. **FOR** task $v_i \in L$
8. **FOR** all computing sensor m_k
9. **IF** $pred(v_i) \not\subseteq T(m_k)$
10. **FOR** $v_n \in pred(v_i) - T(m_k)$
11. **CommTaskSchedule**(v_n,$m(v_n)$,m_k)
12. Assign v_i to m_k, calculate $fit(v_i, m_k, \alpha)$
13. Find m_i^o: $fit(v_i, m_i^o, \alpha) = \min$
14. Keep the schedule with (v,m): $fit(v, m, \alpha) = min$
15. Assign v to m, remove v from L
16. Assign $succ(v)$ on m
17. Update L with any new unassigned mappable tasks
18. Among all schedules with different values of α
19. **IF** $\exists H : length(H) \leq DL$ with $\min energy(H)$
20. Return H
21. **ELSE**
22. Return $H : length(H) = \min$

D. The DVS Algorithm

Due to the discrete nature of task mapping and scheduling, a schedule that meets a deadline may do so with some more slack time until the deadline. The unbalanced load of sensors and the communication scheduling also result in CPU idle time. In the *DVS Phase*, the CPU idle time is exploited by decreasing the CPU speed to reduce computation energy consumption.

Before introducing the DVS Algorithm, we first present the procedure to adjust the CPU speed of a single sensor in a given time interval. A concept of *CPU time utility* η during a time interval $[st, ft]$ is first defined as:

$$\eta = e_{st}^{ft}/(ft - st), \quad \text{(III-D.10)}$$

where e_{st}^{ft} is the CPU execution time during $[st, ft]$. To exploit the CPU slack time, the strategy of the CPU adjustment algorithm is to slow down the CPU

in proportion to the *CPU time utility*. After adjustment, the *CPU time utility* will approach 1 (but smaller than or equal to 1). The CPU speed adjustment algorithm is described in detail as follows:

Input: CPU speed f_{cpu} of sensor m_k in time interval $[st, ft]$;
Output: Adjusted CPU speed f^o_{cpu} and schedule in $[st, ft]$
SpeedAdjust(m_k,st,ft,f_{cpu}):
1. $e^{ft}_{st} \leftarrow 0$, $tt \leftarrow st$
2. **FOR** $v_i \in T^{ft}_{st}(m_k)$ and $v_i \in V$
3. $\quad e^{ft}_{st} \leftarrow e^{ft}_{st} + t_{v_i,m_k}$
4. $\eta \leftarrow e^{ft}_{st}/(ft - st)$
5. $f^o_{cpu} \leftarrow \lceil f_{cpu} \cdot \eta \rceil$
6. **FOR** $v_i \in T^{ft}_{st}(m_k)$ and $v_i \in V$
7. $\quad s_{v_i,m_k} \leftarrow tt$
8. $\quad t_{v_i,m_k} \leftarrow t_{v_i,m_k} \cdot \frac{f_{cpu}}{f^o_{cpu}}$
9. $\quad f_{v_i,m_k} \leftarrow s_{v_i,m_k} + t_{v_i,m_k}$
10. $\quad tt \leftarrow f_{v_i,m_k}$
11. **FOR** $v_i \in T^{ft}_{st}(m_k)$ and $v_i \in R$
12. $\quad$ **IF** $pred(v_i) \in T(m_k)$
13. $\quad\quad s_{v_i,m_k} \leftarrow f_{pred(v_i),m_k}$, $f_{v_i,m_k} \leftarrow f_{pred(v_i),m_k}$
14. Update the energy consumption of m_k

In the algorithm above, the function $\lceil f \rceil$ is a ceiling function that returns the minimum available CPU speed larger than or equal to f.

In the DVS algorithm, the communication tasks on $\mathcal{C}$ are kept unchanged, and their start time and finish time are taken as the upper and lower bound to adjust the corresponding sensors' speed with the **SpeedAdjust** procedure. The DVS algorithm is described in details as follows:

Input: schedule H from the *Mapping and Scheduling Phase*, sensor set SS, application deadline DL
Output: Adjusted schedule H^o
DVS Algorithm:
1. **FOR** sensor $m_k \in SS$
2. $\quad st \leftarrow 0$, $ft \leftarrow \infty$
3. $\quad$ **FOR** tasks $v_i \in T^{\infty}_{st}(m_k)$
4. $\quad\quad$ **IF** There is a copy of v_i: $v^c_i \in T(\mathcal{C})$
5. $\quad\quad\quad$ Find the computation task v_j following v_i
6. $\quad\quad\quad$ **IF** m_k is the sender of v^c_i
7. $\quad\quad\quad\quad ft \leftarrow \min(s_{v^c_i,\mathcal{C}}, s_{v_j,m_k})$
8. $\quad\quad\quad\quad$ **SpeedAdjust**(m_k,st,ft,f^{max}_{cpu})
9. $\quad\quad\quad\quad st \leftarrow ft$
10. $\quad\quad\quad$ **ELSE** /*m_k is the receiver of v^c_i*/
11. $\quad\quad\quad\quad st \leftarrow \max(f_{v^c_i,m_k}, s_{v_j,m_k})$
12. $\quad\quad$ **ELSE IF** v_i is exit-task and $f_{v_i} < DL$
13. $\quad\quad\quad$ **SpeedAdjust**(m_k,st,DL,f^{max}_{cpu})

IV. Simulation Results

The performances of the RT-MapS with the H-CNPT algorithm and the RT-MapS with the H-MinMin algorithm are evaluated through simulations, and denoted as H-CNPT and H-MinMin in this section, respectively. The performance of DCA is also evaluated as a benchmark. DCA is extended such that several sensors perform entry-tasks and send the intermediate results to the cluster head for further processing. DCA algorithm is also implemented with DVS for fair comparison. We run simulations to investigate the following aspects:

- Effect of the application deadline constraints
- Effect of the number of tasks in applications
- Effect of the inter-task dependency
- Evaluation of the energy consumption balance
- Comparison with EbTA [7]

In these simulations, we observe energy consumption, schedule length, and deadline missing ratio (DMR) metrics. 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. The DMR is defined as the ratio of the number of the simulation runs whose schedule length is larger than the imposed deadline over the number of the overall simulation runs.

A. Simulation Parameters

In our simulation study, the bandwidth of the channel is set to 1Mb/s and the transmission range to 10 meters. We assume that there are 10 sensors in a single-hop cluster. Sensors are equipped with the StrongARM SA-1100 microprocessor, whose speed ranges from 59 MHz to 206 MHz with 30 discrete levels. The parameters of Equation II-B.1 - II-B.4 are in coherence with [11], [1], [10] as follows: E_{elec} = 50 nJ/b, ε_{amp} = 10 pJ/b/m^2, 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.

Simulations are run on randomly generated DAGs, which are created based on three parameters: The number of tasks *numTask*, the number of entry-tasks *numEntry*, and the maximum number of predecessors *maxPred*. The number of each non-entry task's predecessors, the computation load, and the resulting data volume of a task are uniformly distributed over [1, *maxPred*], [300K CC, $\pm 10\%$], and [800 bits, $\pm 10\%$], respectively. The simulation results presented in this section correspond to the average of one hundred independent runs.

B. Effect of the Application Deadlines

The effect of application deadlines and DVS adjustment are investigated with randomly generated DAGs as *numTask* = 25, *numEntry* = 6, and *maxPred* = 3. To evaluate

the effect of DVS, the performance of DCA, H-CNPT and H-MinMin before the voltage adjustment (denoted as DCA*, H-CNPT* and H-MinMin*, respectively) are also investigated.

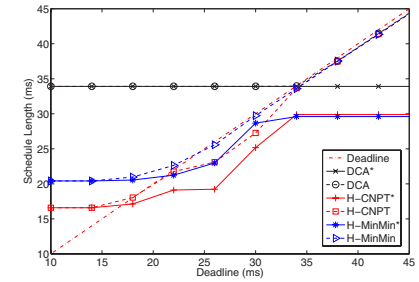
As shown in Fig. 3(a) and Fig. 3(c), both RT-MapS algorithms have better capability to meet small deadlines compared with DCA. When deadlines are very small, though the DMR of RT-MapS algorithms and DCA are all high, the average schedule lengths of RT-MapS are much smaller and closer to deadlines compared with DCA. When deadline increases, the DMR of the RT-MapS algorithms drops much faster than DCA. The better adaptability to deadline changes of RT-MapS algorithms stems from the fact that RT-MapS can have multiple computing sensors in parallel according to deadline constraints, while DCA has only one sensor for high level computing.

Regarding the comparison of the RT-MapS algorithms themselves, both H-CNPT and H-MinMin intend to use less computing sensors to decrease communication energy consumption when the deadline is large, thus their energy consumption and schedule lengths converge. When deadlines are small, H-CNPT outperforms H-MinMin in term of schedule lengths and DMR. The scheduling criteria of H-CNPT is determined by schedule lengths only, while the fitness function of H-MinMin is a combination of schedule length and energy consumption. The tradeoff between schedule length and energy consumption degrades the schedule length performance of H-MinMin algorithm.

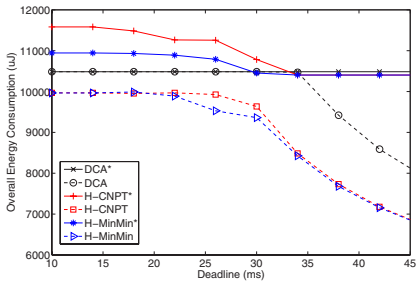
Regarding energy consumption, DCA* has better performance than H-CNPT* and H-MinMin* for most scenarios according to Fig. 3(b). However, by implementing DVS algorithm, this energy consumption difference is significantly reduced. When deadlines are sufficiently large, the DVS adjustment results in 25 - 35 % energy savings by "pushing" the schedule length close to the deadline. Even when deadlines are relatively small and there is little slack time before application deadlines, the DVS adjustment of H-CNPT and H-MinMin can still save about 15% and 10% energy compared with the scenarios without the DVS adjustment, respectively. This energy saving stems from exploiting the slack time caused by the unbalanced load of sensors and communication scheduling. Though the DVS adjustment may increase schedule lengths (Fig. 3(a)), the DMR is not affected (Fig. 3(c)) for any of the simulated deadline values.

C. Effect of the Number of Tasks

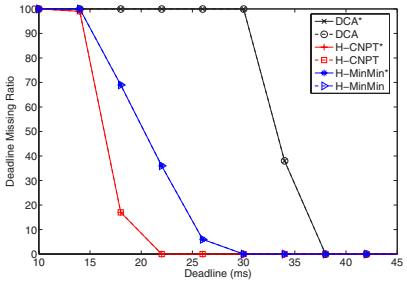
To investigate the effect of number of tasks in applications, three sets of simulations are run on randomly generated DAGs with 20, 25 and 30 tasks (*numEntry* = 6, *maxPred* = 3). According to the simulation results in Fig. 4, energy consumption is dominated by the number of tasks. When the number of tasks increases, the energy consumption of DCA, H-CNPT, and H-MinMin increase



(a) Schedule Length



(b) Energy Consumption



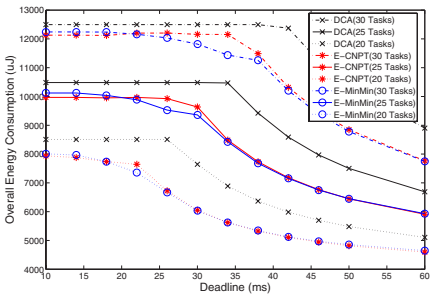
(c) Deadline Missing Ratio

Fig. 3. Effect of Application Deadlines

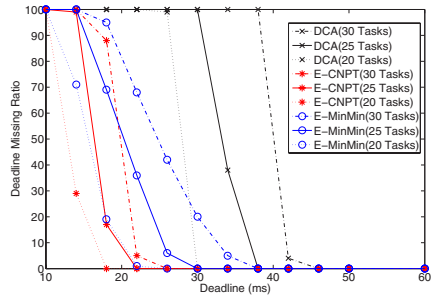
proportionally. Regarding DMR, DCA is affected the most with task volume increment while H-CNPT is affected the least (Fig. 4(b)). Thus, the RT-MapS algorithms have better scalability compared with DCA regarding schedule length and DMR, and H-CNPT outperforms H-MinMin among the two RT-MapS algorithms.

D. Effect of the Inter-Task Dependency

The 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 the simulation results of Fig. 5, the inter-task dependency has almost no effect over the performance of DCA. This is due to the fact that DCA has most of the tasks executed on the cluster head and therefore has the least need

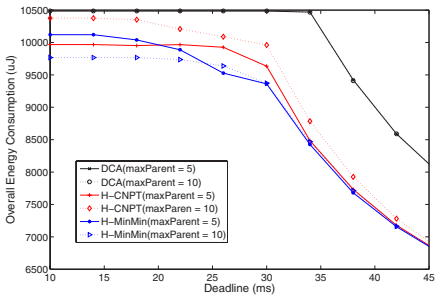


(a) Energy Consumption

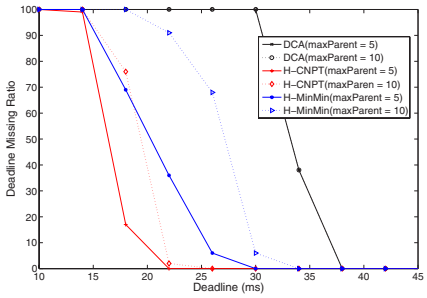


(b) Deadline Missing Ratio

Fig. 4. Effect of Number of Tasks (30 tasks VS 25 tasks VS 20 tasks)



(a) Energy Consumption



(b) Deadline Missing Ratio

Fig. 5. Effect of Inter-task Dependency (25 tasks)

for communication. Regarding H-CNPT and H-MinMin, increasing the in/out degree of DAGs does not introduce new communication tasks but increases the dependency between a result delivery task and its immediate successors. Greater dependency degree between tasks leads to a higher number of communication tasks scheduled on $\mathcal{C}$ and less parallelism between sensors, which leads to more energy consumption and longer schedules, especially with small deadlines. Compared with H-CNPT, H-MinMin is affected more and has a higher possibility of missing deadlines when the communication load increases.

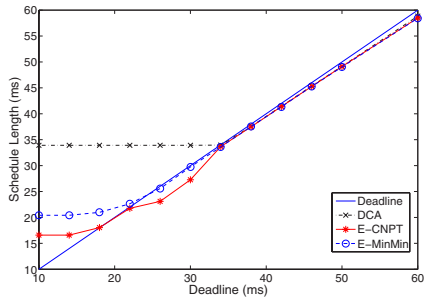
E. Evaluation of the Energy Consumption Balance

The energy consumption balance is another important factor in the WSN design. In this section, the energy consumption balance of the proposed RT-MapS algorithms are evaluated and compared to the DCA algorithm through simulations. The random DAGs considered in the simulations have the parameters of *numTask* = 25, *numEntry* = 6, and *maxPred* = 3. The observed metric is the *Maximum Energy Consumption per Sensor* (MECpS) in addition to the schedule lengths of all sensors. As shown in Fig. 6, when deadlines are small, RT-MapS reduces the execution time by involving more computing sensors, which leads to even distribution of energy consumption among sensors. Thus, both RT-MapS algorithms have relatively small MECpS with small deadlines. On the other hand, DCA assigns the cluster head with all high level task processing, which leads to unbalanced energy consumption and a larger MECpS than RT-MapS. When the deadline increases, the RT-MapS intends to decrease the number of computing sensors to conserve overall application energy consumption. These computing sensors have higher computation load, resulting in increased MECpS for large deadlines. However, when deadlines are sufficiently large, only one sensor will be involved in high level computing in RT-MapS and DCA. The DVS algorithm exploits the slack time before the deadline and further decreases the computing sensor's energy consumption. This DVS incorporation decreases MECpS when deadlines are sufficiently large, as shown in Fig. 6(b).

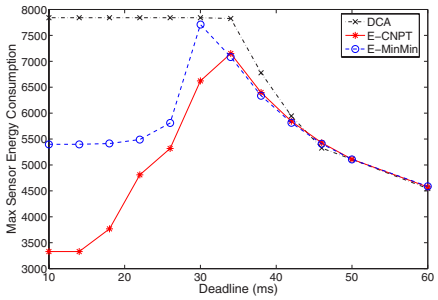
F. Comparison with EbTA

To compare the performance with EbTA[1] , we run simulations on random DAGs with *numTask* = 25, *numEntry* = 6, and *maxPred* = 6 on a cluster with one single-hop wireless channel. Due to the space constraint, we just

[1]The authors would like to thank Yang Yu for providing the simulator of EbTA [7].



(a) Schedule Length



(b) Maximum Energy Consumption per Sensor

Fig. 6. Energy Balance

TABLE I. Comparison with EbTA

Deadline	Metrics	EbTA	H-CNPT	H-MinMin
30ms	Schedule Length (ms)	37.63	26.42	29.96
	DMR (%)	92	0	16
	OAEC (uJ)	11087.1	9905.9	9597.5
	MECpS (uJ)	5658.9	5687.2	7847.8
40ms	Schedule Length (ms)	40.21	39.54	39.45
	DMR (%)	37	0	0
	OAEC (uJ)	10894.7	7544.6	7669.9
	MECpS (uJ)	5469.3	6222.1	6238.2

present partial simulation results in Table I, where each item stands for the average of 100 independent simulation runs. Both RT-MapS algorithms have better capability to meet deadline constraints and are more energy-efficient regarding OAEC compared with EbTA, though the MECpS of EbTA can be smaller with the large deadline. The superior performance of the RT-MapS algorithms mostly stems from the fact that RT-MapS exploits the broadcast feature of wireless channel when scheduling communication events, while a task in EbTA has to send information individually to its immediate successors.

V. Conclusion

In this paper, we propose a real-time task mapping and scheduling (RT-MapS) solution for collaborative in-network processing in DVS enabled WSNs. We consider applications executed in a single-hop cluster of WSNs with deadline constraints. The design objective of RT-MapS is to map and schedule the tasks of an application with the minimum energy consumption subject to the deadline constraint. The wireless channel is modeled as a virtual node to execute communication tasks, and a communication scheduling algorithm is presented with the broadcast leverage feature. Incorporating our communication scheduling algorithm, the modified CNPT and Min-Min algorithm schedule tasks with minimum energy consumption subject to deadline constraints. DVS technique is implemented to further reduce energy consumptions. Simulations show superior performance improvements of RT-MapS compared with existing solutions in terms of guarantee deadline constrains with minimum energy consumption.

References

[1] A. Wang and A. Chandrakasan, "Energy-efficient DSPs for wireless sensor networks," *IEEE Signal Processing Magazine*, pp. 68–78, July 2002.

[2] I. F. Akyidiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks (Elsevier) Journal*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[3] W.-C. Feng, E. Kaiser, W. C. Feng, and M. L. Baillif, "Panoptes: Scalable low-power video sensor networking technologies," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 1, no. 2, pp. 151–167, May 2005.

[4] S. Giannecchini, M. Caccamo, and C.-S. Shih, "Collaborative resource allocation in wireless sensor networks," in *Proc. of Euromicro Conference on Real-Time Systems (ECRTS'04)*, June/July 2004, pp. 35–44.

[5] S. Shivle, R. Castain, H. J. Siegel, A. A. Maciejewski, T. Banka, K. Chindam, S. Dussinger, P. Pichumani, P. Satyasekaan, W. Saylor, D.Sendek, J. Sousa, J. Sridharan, P. Sugavanam, and J. Velazco, "Static mapping of subtasks in a heterogeneous ad hoc grid environment," in *Proc. of Parallel and Distributed Processing Symposium*, Apr. 2004.

[6] Y. Tian, E. Ekici, and F. Özgüner, "Energy-constrained task mapping and scheduling in wireless sensor networks," in *Workshop on Resource Provisioning and Management in Sensor Networks (RPMSN'05), in conjunction with MASS'05*, Nov. 2005.

[7] Y. Yu and V. K. Prasanna, "Energy-balanced task allocation for collaborative processing in wireless sensor networks," *ACM/Kluwer Jourani of Mobile Networks and Applications*, vol. 10, no. 1-2, pp. 115–131, Feb. 2005.

[8] T. Hagras and J. Janecek, "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)*, Oct. 2003, pp. 149–155.

[9] T. D. Braun, H. J. Siegel, N. Beck, L. Boloni, R. F. Freund, D. Hensgen, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, June 2001.

[10] W. B. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[11] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *Proc. of ACM MobiCom'01*, July 2001, pp. 272–286.

[12] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.