Energy-Constrained Task Mapping and Scheduling in Wireless Sensor Networks

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

***Abstract*— Collaboration among sensors through parallel processing mechanisms emerges as a promising solution to achieve high processing power in resource-restricted Wireless Sensor Networks (WSN). Although task mapping and scheduling in wired networks of processors has been well studied in the past, their application to WSNs remains largely unexplored. Due to the limitations of WSNs, existing algorithms cannot be directly implemented in WSNs. In this paper, a task mapping and scheduling solution for energy-constrained applications in WSNs, Energy-constrained Task Mapping and Scheduling (EcoMapS), is presented. EcoMapS incorporates channel modeling, concurrent task mapping, communication and computation scheduling, and sensor failure handling algorithm. The performance of EcoMapS is evaluated through simulations with randomly generated Directed Acyclic Graphs (DAG). Simulation results show significant performance improvements compared with an existing mechanism in terms of minimizing schedule lengths subject to energy consumption constrains.**

I. Introduction

Energy consumption is a fundamental challenge in Wireless Sensor Networks (WSN) due to their unique features such as limited , irreplaceable energy sources and lifetime requirements [1]. Without being able to replace batteries, WSN applications will be imposed with energy consumption constraints to guarantee lifetime requirements. At the same time, many emerging applications require in-network processing with considerable computation demands in these resource-constrained environments. For instance, in target tracking applications [2], sensors collaboratively measure and estimate the location of moving targets or classify targets with an eye toward limited energy consumption. Another set of challenging applications are designed for video sensor networks [3]. Since multimedia processing generally involves computationally intensive operations, computation power emerges as a highly demanding resource in video sensor networks.

In this work we introduce a general solution to provide the computation capacity required by in-network processing subject to energy consumption constraints. WSNs are composed of a large number of relatively low capacity sensors [1]. To provide the demanded computation power for applications, a promising solution is to have sensors collaboratively process information. In [4], [5], [6], collaborative data processing architectures are proposed, where low level tasks are executed on sensing sensors and all other high level processing tasks are offloaded to cluster heads. However, processing high level tasks can still exceed the capacity of cluster heads' computation power. Even when more powerful sensors are introduced as cluster heads [6], overloading "hot spot" sensors with extensive calculations and frequent communications will quickly deplete the sensors' battery, shortening network lifetime. Furthermore, application-specific design of these solutions limit their implementation in generic applications.

Parallel processing among sensors is a promising solution to provide the demanded computation capacity in WSNs. *Task mapping and scheduling* plays an essential role in parallel processing by solving the following problems subject to design objectives: 1. Assignment of tasks onto processing units; 2. Execution sequence of tasks on processing units; 3. Communication schedule between processing units. These problems must also be solved in WSNs to enable in-network processing.

Task mapping and scheduling has been extensively studied in the area of high performance computing [7] [8]. However, traditional parallel processing systems generally assume point-to-point connections between all nodes and communication can occur simultaneously without contention. Thus, task mapping and scheduling solutions in high performance computing cannot be directly implemented in WSNs, and task mapping and scheduling remains largely unexplored in WSNs. Task allocation has been recently discussed in the literature [9] for WSNs. The objective of [9] is to find schedules with balanced energy consumptions for WSN applications. However, the energy-balanced solution in [9] does not consider application energy consumption constraints and cannot provide energy consumption constraint guarantees.

In this paper, we propose a task mapping and scheduling solution for WSNs. We consider energy-constrained applications executed in a single-hop cluster of homogeneous WSNs. *Our proposed solution, **E**nergy-**co**nstrained Task **Map**ping and **S**cheduling (EcoMapS), aims to map and schedule the tasks of an application with minimum schedule length subject to energy consumption constraints*. EcoMapS is application-independent as it is based on the high-level application model that describes the task dependencies through Directed Acyclic Graphs (DAG) (see Section III-B). A channel model is presented for single-hop wireless networks. Based on this channel model, communication scheduling is integrated as part of EcoMapS. In EcoMapS, communication and computation are jointly scheduled in the *Initialization Phase*. In case of sensor failures, a quick recovery algorithm is executed in the *Quick Recovery Phase* to generate an alternative schedule. The quick recovery algorithm is proved to meet energy consumption constraints if applied on an initially feasible solution.

II. Related Work

Task scheduling problem in WSNs has been studied in the literature recently. In [10], an online task scheduling mechanism (CoRAl) is proposed to allocate the network resources between the tasks of periodic applications in WSNs. Upper-bound frequencies of applications are evaluated according to

0-7803-9466-6/05/$20.00 ©2005 IEEE

bandwidth and communication requirements between sensors. The frequencies of the tasks on each sensor are optimized subject to the upper-bound execution frequencies. However, CoRAl does not address mapping tasks to sensor nodes, and energy consumption is not explicitly discussed in [10].

Task mapping mechanisms in wireless networks have been presented in [11], [12]. A TCP-oriented distributed task mapping approach is introduced in [11] for mobile ad hoc networks. A data fusion task mapping mechanism, DFuse, is presented for WSNs in [12]. Both solutions assume an existing underlying network communication mechanism. Communication scheduling between nodes is not addressed explicitly, which has a significant impact on communication efficiency and energy consumption in WSNs.

Task mapping and task scheduling have been jointly considered for mobile computing [13] and WSNs [9] recently. Task mapping and scheduling heuristics are presented in [13] for heterogeneous mobile ad hoc grid environments. However, the communication model adopted in [13] is not well suited for WSNs, which assumes individual channels for each node and concurrent data transmission and reception capacity of every node. In [9], communications over multiple single-hop wireless channels are first modeled as additional linear constraints of an Integer Linear Programming (ILP) problem. Then a heuristic algorithm is presented to provide a practical solution. In [9], energy consumption optimization is addressed through energy-balanced task allocation. However, the energy-balanced solution in [9] does not take energy consumption constraints into account and cannot provide guarantee of application energy consumption constraints.

In this paper, we present a generic task mapping and scheduling solution, EcoMapS, for single-hop clustered wireless sensor networks with the the following salient properties:

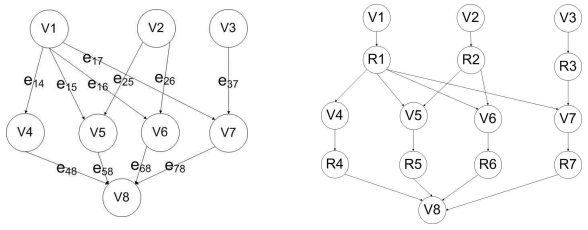
- Task mapping and scheduling are performed jointly.
- Communication and computation are jointly scheduled.
- Based on realistic energy models for computation and communication, EcoMapS aims to provide energy consumption guarantees with minimum schedule lengths.
- Sensor failures are handled at low cost.

III. Preliminaries

A. Network Assumptions

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

- Sensors are grouped into single-hop clusters.
- Each cluster executes an application which is either assigned during the network setup time or distributed by base stations during the network operation. Once assigned, applications are independently executed within each cluster unless new applications arrive. With application arrivals, cluster heads create the schedules for application communication and computation within clusters.
- Time synchronization is locally available within clusters.
- Computation and communication can occur simultaneously on sensor nodes.
- Communication in a cluster is isolated from other clusters through channel-hopping mechanisms such as [14].



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

Fig. 1
DAG and Hyper-DAG Examples

B. Application 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. The weight of a task is represented by the number of CPU clock cycles to execute the task. Given 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. An immediate successor v_j depends on its immediate predecessors such that v_j cannot start execution before it receives results from all of it immediate predecessors. A task without immediate predecessors is called an *entry-task* and a task without immediate successors is called an *exit-task*. A DAG may have multiple entry-tasks and one exit-task. If there are more than one exit-tasks, they will be connected to a pseudo exit-task with computation cost equals zero. Fig. 1(a) shows an example of a DAG, where $V1$, $V2$ and $V3$ are entry-tasks, $V8$ is an exit-task, and $V5$ is the immediate successor and immediate predecessor of $V1$ and $V8$, respectively.

In the DAG scheduling problem, 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 of the 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.

C. Energy Consumption Model

The energy consumptions 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:

$$E_{tx}(l,d) = E_{elec} \cdot l + \varepsilon_{amp} \cdot l \cdot d^2, \qquad \text{(III-C.1)}$$

$$E_{rx}(l) = E_{elec} \cdot l, \qquad \text{(III-C.2)}$$

where E_{elec} and ε_{amp} are hardware parameters [5], [15].

The energy consumption of executing N clock cycles with CPU clock frequency f is given as:

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

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

where V_T is the thermal voltage and C, I_o, n, K and c are processor dependent parameters [4], [5].

It should be noted that the energy consumption model presented above only considers the energy expenditure directly related with application executions, thus energy consumption during idle time is not taken into account.

D. 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 such as energy consumption or schedule length. Let $H^x = \{h_1^x, h_2^x, ..., h_n^x\}$ denote a task mapping and scheduling solution of an application DAG T on a network G, where x is the solution space index. 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 objective of this work is to find an $H^o \in \{H^x\}$ with minimum schedule length under energy consumption constraint, which is formulated as follows:

$$\min length(H^o) = \max_{i,k} f_{i,m_k}; \quad \text{(III-D.5)}$$

$$\text{subject to } energy(H^o) = \sum_{i,k} c_{i,m_k} \leq EB, \quad \text{(III-D.6)}$$

where $length(H)$ and $energy(H)$ are the schedule length and energy consumption of schedule H, respectively, and EB is the energy consumption constraint (referred to as *Energy Budget*). DAG scheduling problem is shown to be an NP-complete problem in general [16]. Hence, 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 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 $[st, ft]$.

IV. The Proposed EcoMapS Algorithm

Our proposed EcoMapS has two phases: *Initialization Phase* and *Quick Recovery Phase*. In the *Initialization Phase*, tasks are assigned to sensors, the execution sequence of tasks are decided, and communications between sensors are scheduled with respect to the *Dependency Constraint*. We extend and implement the low-complexity list-scheduling algorithm, CNPT [8] for task mapping and scheduling in this phase. The objective of the Extended CNPT (E-CNPT) algorithm is set to minimize scheduling lengths subject to energy consumption constraints. Since sensors are prone to failures, a quick recovery algorithm is designed for the *Quick Recovery Phase* to handle runtime sensor failures. The scheduling algorithms above are executed on cluster heads when applications are assigned to clusters. In case of a loss of a cluster head, a new cluster head is selected via the clustering algorithm in use, and schedules will be regenerated by the new cluster head.

In the following sections, we present main components of our proposed EcoMapS algorithm, namely, *Wireless channel modeling and Hyper-DAG extension, Communication scheduling algorithm, E-CNPT algorithm, and Quick Recovery algorithm*. E-CNPT is based on the CNPT algorithm and executed during the Initialization Phase. The original CNPT algorithms is designed for traditional parallel processing with the assumption of point-to-point connections among all nodes. To extend CNPT for WSNs, we developed a *communication scheduling algorithm* based on the *wireless channel model* and the *Hyper-DAG representation* of applications. The communication scheduling algorithm is embedded in the execution of E-CNPT to satisfy the *Dependency Constraint*. In case of sensor failures, the schedules generated in the Initialization Phase will be adjusted with the *Quick Recovery Algorithm* instead of rescheduling with the E-CNPT algorithm, which can be time consuming. The E-CNPT algorithm will be executed only when the performance degrades to a certain threshold, a new application arrives, or cluster heads fail.

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. Hence, the wireless channel is modeled as a virtual node $\mathcal{C}$ that executes one communication task at any time instance. Hence, a cluster can be modeled as a star-networ,k where all sensors only have connections with the virtual node $\mathcal{C}$ [10]. The communication latency between sensor nodes and $\mathcal{C}$ can be considered zero as all wireless communications are accounted for by the tasks executed on $\mathcal{C}$. Assuming that the cluster has p sensors $M = \{m_k\}$ $(0 \leq k < p)$, 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 links between nodes of M'.

To implement this channel model, communication events between computation tasks 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. The weight of R_i equals to the result 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, 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 III-B is rephrased as follows: in the Hyper-DAG scheduling problem, 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 scheduled on the same node. It should be noted that the channel model presented above assumes a single-hop clustered environment. However, this model can be generalized to multi-hop networks by taking the inference avoidance into consideration. We will defer the discussion of multi-hop channel modeling to our future work.

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. Thus, we present our communication scheduling algorithm in this section. As shown in Section IV-C, communication scheduling algorithm is integrated into our Hyper-DAG task mapping and scheduling algorithm E-CNPT. Based on the Hyper-DAG and the channel model presented in Section IV-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 unicasts between the sender and the multiple receivers. The communication scheduling algorithm is presented below.

Input: Communication task v_i; sender m_s; 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 $v_i \in T(m_s)$
4. Find time interval [st,ft]:
5. $T_{st}^{ft}(\mathcal{C}) = \emptyset$, $ft - st \geq t_{v_i,\mathcal{C}}$
6. $st \geq f_{v_i,m_s}$, $st = \min$
7. Schedule a copy of v_i to $\mathcal{C}$:
8. $v_i^c \in T(\mathcal{C})$, $s_{v_i^c,\mathcal{C}} \leftarrow st$
9. Update the energy consumption of m_s
10. Schedule a copy of v_i^c to m_r:
11. $v_i^r \in T(m_r)$, $s_{v_i^r,m_r} \leftarrow f_{v_i^c,\mathcal{C}}$
12. Update the energy consumption of m_r
13. Return
14. **ELSE**
15. Schedule a copy of v_i^c to m_r:
16. $v_i^r \in T(m_r)$, $s_{v_i^k,m_r} \leftarrow f_{v_i^c,\mathcal{C}}$
17. Update the energy consumption of m_r
18. Return

In the algorithm above, Step 2-12 stands for originating a new communication from m_s to m_r, and Step 15-17 represents reception of a broadcast data without interference. Compared with originating a new communication, the broadcast reception method leads to energy saving of one data transmission for each additional data reception. Under our communication scheduling algorithm, one data transmission may reach multiple receivers. Effectively, we achieve $multicast$ distribution of data, which has been proved to be energy efficient.

C. Task Mapping and Scheduling with E-CNPT Algorithm

In the *Initialization Phase* of EconMapS, the tasks of Hyper-DAGs are mapped and scheduled on sensors. During task mapping, several constraints have to be satisfied: a computation task can be assigned only on sensor nodes, a communication task can be assigned on sensors and $\mathcal{C}$ with multiple copies, and a communication task assigned on a sensor node has zero execution length and energy cost. These constraints together with the *Dependency Constraint* are given as follwos:

- $\forall v_i \in R : t_{i,m_k} = 0, c_{i,m_k} = 0$, when $m_k \neq \mathcal{C}$
- $\forall v_i \in V : t_{i,\mathcal{C}} = \infty, c_{i,\mathcal{C}} = \infty$
- 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 [16] and heuristic algorithms are needed to obtain practical solutions. Because of its satisfying performance with relatively low complexity, CNPT algorithm [8] is extended and implemented in the Initialization Phase of EcoMapS, and is denoted as E-CNPT.

The objective of E-CNPT is set to minimize schedule lengths subject to energy consumption constraints. The strategy of E-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 schedule length under the energy consumption constraint, the design objective of E-CNPT is achieved. Here, a *computing sensor* is a sensor that can execute non-entry tasks in addition to entry-tasks. Similar to CNPT, E-CNPT also has two stages: *listing stage* and *sensor assignment stage*. In the *listing stage*, tasks will be sequentialized into a queue L in the order that the most critical path comes the first and a task is always enqueued after its 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 one schedule is chosen as the solution according to the design objective. The *listing stage* and *sensor assignment stage* of E-CNPT are introduced individually as follows.

1) Listing Stage: The Listing Stage of E-CNPT is similar to that of CNPT [8]. In the Listing Stage of E-CNPT, the Earliest Start Time $EST(v_i)$ of task v_i is first calculated for each vertex by traversing the Hyper-DAG downward from the entry-tasks to the exit-task. The Latest Start Time $LST(v_i)$ of task v_i is then calculated in the reverse direction. During the calculation, the entry-tasks have *EST* = 0 and the exit-task has *LST* = *EST*. *EST* and *LST* are calculated as follows:

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

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

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 $v_i \in E$. Then, the Critical Nodes (CN) are pushed into the stack *S* in the decreasing order of their *LST*. Here, a CN vertex is a vertex with the same value of *EST* and *LST*. Consequently, if *top(S)* has un-stacked immediate predecessors, the immediate predecessor with the minimum *LST* is pushed into the stack; otherwise, *top(S)* is popped and enqueued into a queue *L*. The Listing Phase ends when the stack is empty. After the Listing Phase, the task graph is sequentialized into *L* and is ready for the Sensor Assignment Phase. It should be noted that the EST and LST are for the purpose of evaluating the critical path of a Hyper-DAG, and do not represent the actual execution start time of tasks.

2) Sensor Assignment Stage: In the Sensor Assignment Stage, E-CNPT will iteratively search the schedule space with different number of computing sensors. Among these schedules, the one with the minimum schedule length under the energy consumption constraint is chosen as the solution. If no schedule meetings the energy constraint, the best effort is made by choosing the one with the minimum energy consumption. The E-CNPT algorithm is given below.

Input: Task queue L ; number of available sensors in the cluster p; energy budget EB
Output: Schedule H^o of tasks in L with minimum schedule length under energy budget constraint
E-CNPT Algorithm:
1. $L^o \leftarrow \infty$ /*optimal schedule length*/
2. $E_{min} \leftarrow \infty$ /*minimum energy consumption*/
3. **FOR** q = 1 to p
4. H = SingleCNPT(L,q)
6. **IF** $energy(H) < E_{min}$
7. $E_{min} \leftarrow energy(H)$
8. $H_{min} \leftarrow H$ /*schedule with minimum energy consumption*/
9. **IF** $energy(H) \leq EB$ and $length(H) < L^o$
10. $L^o \leftarrow length(H)$
11. $H^o \leftarrow H$ /*optimal schedule*/
12. **IF** $E_{min} \leq EB$
13. return H^o
14. **ELSE**
15. return H_{min}

In the E-CNPT algorithm above, SingleCNPT(L,q) is a single round of task scheduling that schedules the tasks in L with q computing sensors. It should be noted that the parameter q is just the upper bound of the number of computing sensors that can be involved. The actual number of computing sensors can be smaller than q depending on applications and scheduling algorithms. 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* has to be satisfied via communication scheduling. SingleCNPT(L,q) is described as follows.

Input: task queue L; number of computing sensors q
Output: Assignment of tasks in L
SingleMapSchedule:
while L is not empty
1. Dequeue γ_i from L
2. **IF** $\gamma_i \in R$ /* communication task */
3. Assign γ_i to node $m(pred(\gamma_i))$
4. **ELSE IF** $pred(\gamma_i) = \emptyset$ /*entry-tasks*/
5. Assign γ_i to node m_i^o with min $EAT(m_i^o)$
6. **ELSE** /* non-entry computation tasks*/
7. **FOR** all computing sensors $\{m_k\}$, calculate EEST(γ_i, m_k):
8. **IF** $pred(\gamma_i) \subseteq T(m_k)$
9. EEST(γ_i, m_k) $\leftarrow \max(EAT(m_k), f_{pred(\gamma_i),m_k})$
10. **ELSE** /*communication between sensors is needed*/
11. **FOR** $\gamma_n \in pred(\gamma_i) - T(m_k)$
12. CommTaskSchedule(γ_n,$m(\gamma_n)$,m_k)
13. EEST(γ_i, m_k) $\leftarrow \max(EAT(m_k), f_{pred(\gamma_i),m_k})$
14. keep the scheme with minimum (EEST(γ_i, m^o))
15. schedule γ_i on m^o accordingly

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

D. Sensor Failure Handling with Quick Recovery Algorithm

In WSNs, sensors are prone to failures. In case of sensor failures, the schedule created by the E-CNPT in the Initialization Phase will not be a feasible solution. For such a situation, the WSN's functionality needs to be recovered as soon as possible. Instead of rescheduling from scratch, which can be time consuming, a low-complexity recovery algorithm is presented in this section to quickly recover sensor failures. The E-CNPT algorithm is not executed unless the performance of the recovered schedule degrades to certain threshold.

The basic idea of the quick recovery algorithm is to merge the tasks of the failing sensor onto one of the other working sensors that has the most idle time, and shift all other potentially affected tasks accordingly. If there are more than one sensor failures, the quick recovery algorithm is iteratively executed to handle the failures one by one. The rationale behind merging the tasks of the failing sensor onto another sensor instead of re-distributing the tasks among all of the working sensors is to guarantee the energy consumption constraint, as proved in *Theorem IV-D.1*.

Input: The failing sensor m_f, original sensor set SS, original mapping and scheduling scheme H^o
Output: Recovered schedule H^s on the working sensors
quickRecovery:
1. **IF** $\exists m_{idle} \in SS - \{m_f\} : T(m_{idle}) = \emptyset$
2. reassign T(m_f) onto m_{idle}
3. **ELSE** /*no free sensor, have to merge the tasks*/
4. find $m^o \in SS - \{m_f\}$: IR(m^o) = minimum
5. $t \leftarrow 0$, $\Delta t \leftarrow 0$ /*task adjustment parameters*/
6. **FOR** $v_i \in S = T(m_f) \cup T(m^o)$ with minimum start time
7. **IF** $v_i \in V$ /*computation task*/
8. schedule v_i onto m^o
9. **ELSE** /*communication task*/
10. **IF** there is a duplicated copy of v_i in S
11. remove the duplicated copy
12. find the copy of v_i on $\mathcal{C}$: v_i^c
13. **IF** m_f and m^o are the only sender and receiver of v_i^c
14. remove v_i^c from $\mathcal{C}$
15. **ELSE IF** $pred(v_i) \in S$ /*deliver result to other tasks*/

16. schedule v_i right after $pred(v_i)$
17. **IF** $succ(v_i) \not\subseteq S$ /*may affect tasks on other sensors*/
18. find the copy of v_i on $\mathcal{C}$: v_i^c
19. **IF** $f_{v_i,m^o} > s_{v_i^c,\mathcal{C}}$
20. $\Delta t \leftarrow max(\Delta t, f_{v_i,m^o} - s_{v_i^p,\mathcal{C}})$
21. $t' \leftarrow f_{v_i^c,\mathcal{C}}$
22. **FOR** $m_l \in SS \cup \{\mathcal{C}\} - \{m_f, m^o\}$
23. postpone unadjusted $\gamma_j \in T_t^{t'}(m_l)$ by Δt
24. t $\leftarrow t'$
25. **ELSE** /*receive result from tasks on other sensors*/
26. find the copy of v_i on $\mathcal{C}$: v_i^c
27. $t' \leftarrow f_{v_i^c,\mathcal{C}}$
28. **FOR** $m_l \in SS \cup \{\mathcal{C}\} - \{m_f, m^o\}$
29. postpone unadjusted $\gamma_j \in T_t^{t'}(m_l)$ by Δt
30. t $\leftarrow t'$
31. postpone all unadjusted tasks by Δt
32. **IF** $\frac{length(H^s)}{length(H^o)} > TH$
33. call *Initialization Algorithm*

In the algorithm above, IR(m_k) is the idle time ratio of sensor m_k, and TH is the threshold of unacceptable performance degrade in quick recovery.

Theorem IV-D.1: The recovered schedule H^s meets the energy consumption budget constraint, i.e., $energy(H^o) \leq EB \Rightarrow energy(H^s) \leq EB$

Proof: The energy consumption of a schedule H is composed of computation energy ($compEng(H)$) and communication energy ($commEng(H)$). Since $compEng(H)$ is fixed for an application in homogeneous WSNs, $compEng(H^s) = compEng(H^o)$ holds. $commEng(H)$ is determined by the communication tasks assigned on $\mathcal{C}$. According to Step 14, 23, and 29 of the *quickRecovery* algorithm, the only operations related with the communication tasks on $\mathcal{C}$ are task removals and task shifting in time domain. In other words, no new tasks are assigned to $\mathcal{C}$ and, therefore, no additional energy is consumed for communication. Hence, $commEng(H^s) \leq commEng(H^o)$ holds. If $energy(H^o) \leq EB$, then $energy(H^s) = compEng(H^s) + commEng(H^s) \leq compEng(H^o) + commEng(H^o) = energy(H^o) \leq EB$ holds, as well. ■

E. Computation Complexity

Assume that the application T is represented as $T = (V, E)$, $|V| = v$, $|E| = e$, the number of entry-tasks is f, and the cluster has p sensor nodes. For E-CNPT, the Hyper-DAG is $T' = (V', E')$, where $|V'| = 2v$ and $|E'| = 2e$. The time complexity of EcoMapS is analyzed as follows.

- Listing Stage of E-CNPT: similar to CNPT [8], the complexity is $O(v + e)$.
- SingleMapSchedule of E-CNPT: the communication tasks have complexity of $O(v)$, the entry-tasks have complexity of $O(fp)$, other non-entry computation tasks have complexity of $O((v-f) \cdot p \cdot e/v)$. Hence, the complexity of SingleMapSchedule is $O(v + fp + (v-f) \cdot p \cdot e/v)$. For the worst case, $e = O(v^2)$ and $f = O(v)$, thus the complexity of SingleMapSchedule is $O(pv^2)$ for the worst case.
- Sensor Assignment Stage of E-CNPT: Sensor Assignment Stage has the complexity as $O(v^2p^2)$ for the worst case.
- E-CNPT aglorithm of EcoMapS: considering the factors above together, the worst case complexity is $O(v^2p^2)$ + $O(v + e)$ = $O(v^2p^2)$.

Regarding the Quick Recovery Algorithm of EcoMapS, all tasks including communication tasks and computation tasks will be adjusted at most once. So, the complexity for the Quick Recovery Algorithm is $O(2v - 1) = O(v)$.

V. Simulation Results

The performance of EcoMapS algorithm is evaluated through simulations. The performance of the distributed computation architecture of [5] [6] (referred to as 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. We have run simulations to investigate the following aspects:

- Effect of energy consumption constraints
- Effect of the number of tasks in applications
- Effect of the inter-task dependency
- Performance of the Quick Recovery Algorithm

In these simulations, we observe energy consumption and schedule length 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 presented simulation results correspond to the average of five hundred independent runs.

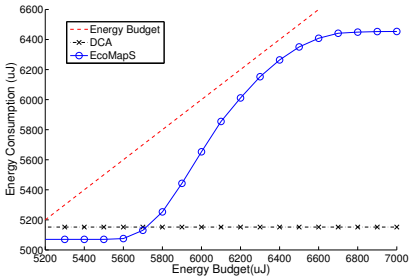
A. Simulation Parameters

In our simulation study, the bandwidth of the channel is set to 1Mb/s and the transmission range r = 10 meters. We assume 10 sensors in a cluster. The sensors are equipped with the StrongARM SA-1100 microprocessor with the CPU frequency be 100 MHz. The parameters of Equation III-C.1 - III-C.4 are in coherence with [4], [5], [15] 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, 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 (in units of kilo-clcok-cycle, KCC), and the resulting data volume (in units of bit) of a task are uniformly distributed over [1, *maxPred*], [300K CC $\pm 10\%$], and [800 bits $\pm 10\%$], respectively.

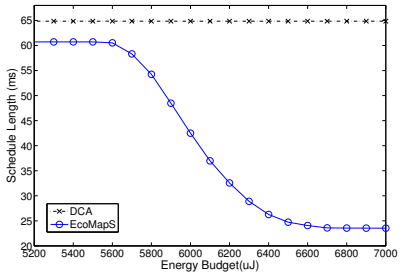
B. Effect of the Energy Consumption Constraints

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 Energy Budget.

As shown in Fig. 2, EcoMapS has a better capability to adjust its schedule according to energy budget compared with DCA. When the energy budget is small, EcoMapS converges to use one sensor for computation. Instead of sending all sensed data to cluster heads, EcoMapS chooses one of the sensing sensor for computation, which saves energy and shortens schedule lengths. When energy budget increases,

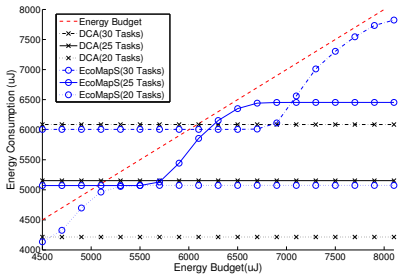


(a) Energy Consumption

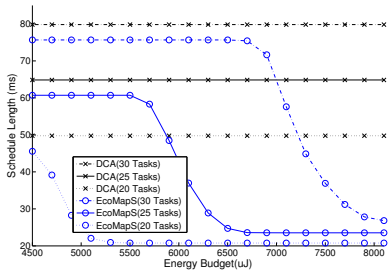


(b) Schedule Length

Fig. 2
EFFECT OF ENERGY BUDGET



(a) Energy Consumption



(b) Schedule Length

Fig. 3
EFFECT OF NUMBER OF TASKS

EcoMapS has more sensors involved in computation, which decreases schedule lengths at the cost of larger energy consumption. On the other hand, DCA cannot adjust its schedule to availability of more energy. Compared with DCA, EcoMapS can lead to about 59% schedule length improvement.

C. Effect of the Number of Tasks in Applications

To test 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 and schedule length of DCA increase proportionally. On the other hand, EcoMapS adapts itself to the increasing energy budget. For the extreme scenarios with small and large energy budgets, the schedule lengths and energy consumption of EcoMapS increase in proportion to the increment of the number of tasks. For the scenarios in the middle, EcoMapS adapts its schedule lengths and energy consumptions according to the available energy budget when the number of tasks increases.

D. Effect of the Inter-task Dependency

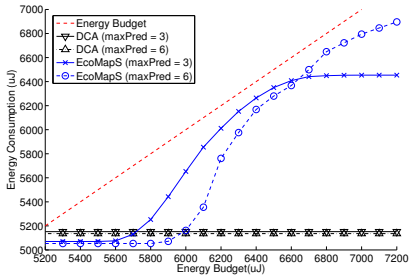
The inter-task dependency is determined by the in/out degree of application DAGs. Simulations with sets of DAGs with $maxPred = 3$ and $maxPred = 6$ (*numTask* = *25*, *numEntry* = 6) are executed. According to the simulation results of Fig. 4, the inter-task dependency has almost no effect over the performance of DCA. The robustness of DCA against inter-task dependency changes stems from the fact that inter-task dependency affects communication scheduling, and DCA has most of the tasks executed on the cluster head with less needs for communication. For EcoMapS, increasing the in/out degree of DAGs does not introduce new communication tasks in the Hyper-DAG, but increases the dependency between a communication task and its immediate successors. Larger dependency between tasks leads to a higher number of communication tasks scheduled on $\mathcal{C}$ and less parallelism in execution, leading to more energy consumption and longer schedules. Though the performance of EcoMapS degrades with higher inter-task dependency, EcoMapS still outperforms DCA with respect to schedule lengths (Fig. 4(b)).

E. Performance of the Quick Recovery Algorithm

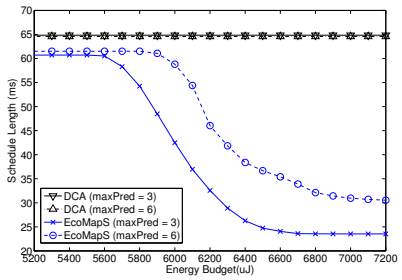
The Quick Recovery Algorithm is evaluated in this section. Since the recovery mechanism with idle senors as backup is trivial, the tested scenarios only consider task merging cases without idle sensors. The simulated scenarios are generated by randomly selecting one failing sensor and merging its tasks onto other working sensors. From Fig. 5(a), we can see that as long as the original schedule meets energy consumption constraints, the recovered schedule satisfies the constraint as well. As we discussed in the proof of Theorem IV-D.1, task merging leads to less energy consumptions with the cost of longer schedule lengths according to Fig. 5(b).

VI. CONCLUSION

Parallel processing among sensors is a promising solution to provide the computation capacity required by in-network

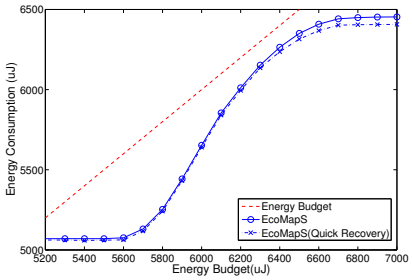


(a) Energy Consumption

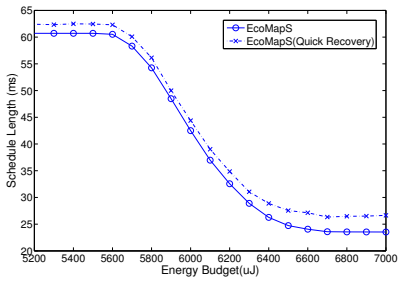


(b) Schedule Length

Fig. 4
EFFECT OF INTER-TASK DEPENDENCY



(a) Energy Consumption



(b) Schedule Length

Fig. 5
PERFORMANCE OF QUICK RECOVERY

processing. In this paper, we present a task mapping and scheduling solution, namely, Energy-constraint Task Mapping and Scheduling (EcoMapS), for one-hop clustered homogeneous WSNs. The objective of EcoMapS is to minimize schedule lengths of applications under energy consumption constraints. Using a new channel model and communication scheduling algorithm, the extended CNPT algorithm schedules tasks subject to the design objective. A quick recovery algorithm is proposed to handle sensor failures. Simulations with randomly generated DAGs show that EcoMapS provides superior performance when compared with DCA. Our future work includes extending our wireless channel model to multi-hop clusters of wireless sensor networks and extending our solutions to heterogeneous WSNs.

REFERENCES

[1] I. F. Akyidiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Elsevier Computer Networks Journal*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[2] T. Vercauteren, D. Guo, and X. Wang, "Joint multiple target tracking and classification in collaborative sensor networks," *IEEE Journal on Selected Areas in Communication*, vol. 23, no. 4, pp. 714–723, Apr. 2005.

[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] 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.

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

[6] R. Kumar, V. Tsiatsis, and M. B. Srivastava, "Computation hierarchy for in-network processing," in *Proc. of the 2nd ACM international conference on Wireless Sensor Networks and Applications (WSNA'03)*, Sept. 2003, pp. 68–77.

[7] A. Dogan and F. Özgüner, "Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogenous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 308–323, Mar. 2002.

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

[10] 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.

[11] P. Basu, W. Ke, and T. D. C. Little, "Dynamic task-based anycasting in mobile ad hoc networks," *Mobile Networks and Applications*, vol. 8, no. 5, pp. 593–612, Oct. 2003.

[12] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran, "DFuse: A framework for distributed data fusion," in *Proc. of The ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, Nov. 2003, pp. 114–125.

[13] 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.

[14] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks," in *Proc. of ACM MobiCom'04*, Sept./Oct. 2004, pp. 216–230.

[15] 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.

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