WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2007; **7**:893–907
Published online 10 May 2007 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/wcm.502

Cluster-based information processing in wireless sensor networks: an energy-aware approach‡

Yuan Tian[1*,†], Eylem Ekici[2] and Füsun Özgüner[2]
[1]*Bosch Research and Technology Center North America, Palo Alto, CA, U.S.A.*
[2]*Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, U.S.A.*

Summary

Emerging Wireless Sensor Network (WSN) applications demand considerable computation capacity for in-network processing in resource limited WSN environments. To achieve the required processing capacity under energy consumption constraints, collaboration among sensors through parallel processing methods emerges as a promising solution. Although such methods have been extensively studied in wired networks of processors, its counterpart for WSNs remains largely unexplored. In this paper, a localized task mapping and scheduling solution for energy-constrained applications in WSNs, Energy-constrained Task Mapping and Scheduling (EcoMapS), is presented. EcoMapS guarantees energy consumption constraints while minimizing schedule length. EcoMapS incorporates channel modeling, concurrent task mapping, communication and computation scheduling, and sensor failure handling algorithms. Simulation results show significant performance improvements of EcoMapS over existing mechanisms in terms of minimizing schedule lengths subject to energy consumption constrains. Copyright © 2007 John Wiley & Sons, Ltd.

KEY WORDS: wireless sensor network; cluster; in-network processing; task mapping and scheduling

1. Introduction

Wireless Sensor Networks (WSNs) are envisioned to observe large and inhospitable environments at close ranges for extended periods of time. WSNs are generally composed of a large number of sensors with relatively low computation capacity and limited energy supply [1]. Many emerging WSN applications require computationally expensive processing operations to be performed in the network. For instance, in-target tracking applications [2], sensors collaboratively measure, track, and classify moving targets. Operations such as Bayesian Estimation and data fusion must be executed in the WSN. In-network processing is essential for energy-efficiency in WSNs [1], especially in video sensor networks [3]. A simple motivating example is shown in Figure 1, where four calibrated camera sensors collaboratively detect an intruder's location.

The sensors first collaboratively estimated the intruder's position, which drastically reduces data volume by several orders of magnitude. Thus, it is

*Correspondence to: Yuan Tian, Bosch Research and Technology Center North America, 4009 Miranda Ave., Palo Alto, CA, U.S.A.
†E-mail: Yuan.Tian@us.bosch.com

‡A preliminary version of this paper has appeared in RPMSN 2005.

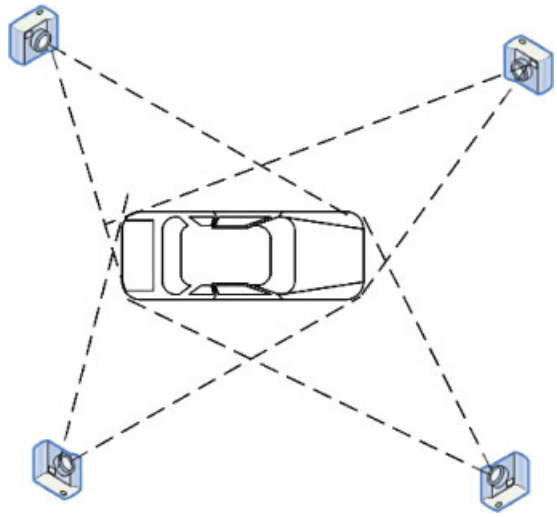


Fig 1. A simplified distributed video surveillance example.

more energy-efficient to send the processed data than delivering the raw data in large-scale WSNs, where base stations can be multiple hops away. However, such multi-media applications as distributed visual surveillance [4] may demand considerable computation power that is beyond the capacity of each individual sensor. Furthermore, limiting energy consumption while performing these operations is of vital importance to prolong network lifetime. Thus, it is desirable to develop a general solution to provide the computation capacity required by in-network processing subject to energy consumption constraints. A promising solution is to have sensors collaboratively process information. *Task mapping and scheduling* [5] plays an essential role in parallel processing by solving the following problems subject to certain design objectives:

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

Task mapping and scheduling has extensively been studied in the area of high performance computing [5–7]. However, existing solutions for wired networks cannot directly be implemented in WSNs since the wireless communication scheduling is not addressed. Furthermore, these solutions do not explicitly consider energy consumption during communication and task execution, which is one of the major constraints 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. System level optimization is achieved through collective result of local optimizations.

In this paper, we propose a localized task mapping and scheduling solution for WSNs. We consider energy-constrained applications executed in a single-hop cluster of a homogeneous WSN. Our proposed solution, Energy-constrained Task Mapping and Scheduling (EcoMapS), aims to map and schedule the tasks of an application with the minimum schedule length subject to energy consumption constraints. EcoMapS is based on the high-level application model that describes the task dependencies through Directed Acyclic Graphs (DAG) [8]. Therefore, EcoMapS can be used with arbitrary applications. In EcoMapS, communication and computation are jointly scheduled in the *Initialization Phase*. In case of sensor failures, replacement schedules are computed in the *Quick Recovery Phase*. We prove that the quick recovery algorithm meets energy consumption constraints if applied on an initially feasible solution.

2. Related Work

Localized task mapping and scheduling problems in WSNs have been studied in the literature recently. In Reference [9], an online task scheduling mechanism (CoRAl) is proposed to iteratively allocate the network resources between the tasks of periodic applications in WSNs: The upper-bound frequencies of applications are first evaluated according to the bandwidth and communication requirements between sensors. The frequencies of the tasks on each sensor are then optimized subject to the upper-bound execution frequencies. However, CoRAl does not address mapping tasks to sensor nodes. Furthermore, energy consumption is not explicitly discussed in Reference [9].

Distributed Computing Architecture (DCA) is presented in Reference [10], 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' computation power. Furthermore, application-specific design of DCA limits its implementation for generic applications.

Localized task mapping and task scheduling have been jointly considered for real-time applications in mobile computing [7] and WSNs [8] recently. Task

DOI: 10.1002/wcm

mapping and scheduling heuristics are presented in Reference [7] for heterogeneous mobile ad hoc grid environments. However, the communication model adopted in Reference [7] is not well-suited for WSNs, which assumes individual channels for each node and concurrent data transmission and reception capacity of every node. In Reference [8], Energy-balanced Task Allocation (EbTA) is presented to minimize balanced energy consumption subject to application deadline constraints. In Reference [8], communications over multiple wireless channels are modeled as additional linear constraints of an Integer Linear Programming (ILP) problem. Then a heuristic algorithm is presented to provide a practical solution. Furthermore, Dynamic Voltage Scaling (DVS) mechanism is implemented to conserve energy. However, the communication scheduling model in Reference [8] does not exploit the broadcast nature of wireless communication, which can reduce energy consumption and execution time. The energy-balanced solution of Reference [8] does not take energy consumption constraints into account and cannot provide energy consumption guarantees.

Different from the work above, in this paper, we present a generic task mapping and scheduling solution, EcoMapS, for single-hop clustered WSNs with the following salient properties:

- Task mapping and task scheduling are considered simultaneously.
- The single-hop wireless channel is modeled as a virtual node, and applications are represented to reflect the broadcast nature of wireless communication.
- Communication and computation events are jointly scheduled.
- Based on realistic energy models, EcoMapS aims to provide energy consumption guarantees with minimum schedule lengths.
- A quick recovery mechanism is designed to handle sensor failures.

3. Preliminaries

3.1. Network Assumptions

The following assumptions are made regarding the WSN:

- Each cluster executes an application which is either assigned during the network setup time or remotely 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 execution within clusters.
- Sensors are synchronized within clusters.
- Computation and communication can occur simultaneously on sensor nodes as supported by various platforms including MICA2DOT running TinyOS.
- Communication within a cluster is isolated from other clusters through time division or channel hopping mechanisms such as described in Reference [11] with appropriate hardware support (eg., Chipcon CC2420).

It should be noted that while the intra-cluster communication is isolated from other clusters, communication across clusters is assumed to be handled over common time slots or channels orthogonal to those used inside a cluster. The cooperative transmission mechanism [12] can also be implemented to avoid interference. As such, information flow across the network is not hindered by intra-cluster communication isolation.

3.2. Application Model

To have an application-independent solution, applications are represented by DAGs. 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. E contains directed edges e_{ij} for each task $v_i \in V$ that task $v_j \in V$ depends on. Given an edge e_{ij}, v_i is called the immediate predecessor of v_j, and v_j is the immediate successor of v_i. 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. Figure 2 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.

DOI: 10.1002/wcm

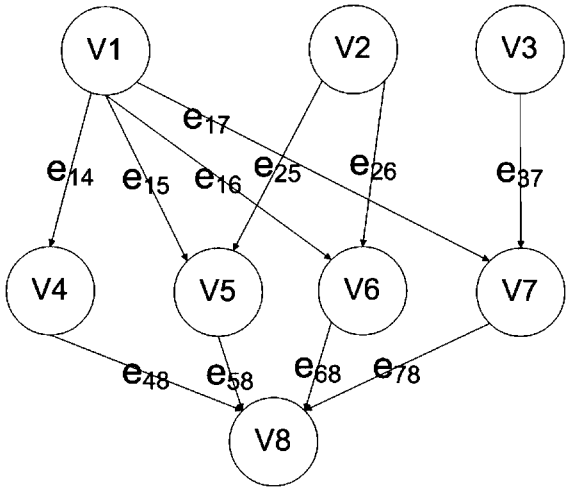


Fig 2. An example DAG.

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

3.3. 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 $E_{\text{tx}}(l, d)$ and $E_{\text{rx}}(l)$, respectively:

$$E_{\text{tx}}(l, d) = E_{\text{elec}} \cdot l + \varepsilon_{\text{amp}} \cdot l \cdot d^2, \quad E_{\text{rx}}(l) = E_{\text{elec}} \cdot l, \tag{1}$$

where E_{elec} and ε_{amp} are hardware related parameters [10,13]. The energy consumption of executing N clock cycles with CPU clock frequency f is given as:

$$E_{\text{comp}}(V_{dd}, f) = NCV_{dd}^2 + V_{dd}\left(I_o e^{\frac{V_{dd}}{nV_T}}\right)\left(\frac{N}{f}\right), \quad f \simeq K(V_{dd} - c) \tag{2}$$

where V_T is the thermal voltage and C, I_o, n, K and c are processor-dependent parameters [10,14]. 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. However, our computation and communication schedules can also be utilized as sensor sleep schedules, where sensors go to sleep when there are no scheduled communication and computation events.

3.4. Problem Statement

In general, a task mapping and scheduling problem is defined as calculating 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 $H^x = \{h_1^x, h_2^x, \ldots, h_n^x\}$ 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 $(v_i, m_k, s_{v_i,m_k}, t_{v_i,m_k}, f_{v_i,m_k}, c_{v_i,m_k})$, where m_k represents the node to which task v_i is assigned, s_{v_i,m_k} and f_{v_i,m_k} represent the start time and finish time of v_i, and t_{v_i,m_k}, and c_{v_i,m_k} represent the execution length and energy consumption of v_i on node m_k, respectively. The design objective of our proposed EcoMapS solution is to find an $H^o \in \{H^x\}$ that has the minimum schedule length subject to energy consumption constraints, which can be formulated as follows:

$$\min \text{length}(H^o) = \max_{i,k} f_{v_i,m_k}, \text{ subject to}$$
$$\text{energy}\ (H^o) = \sum_{i,k} c_{v_i,m_k} \leq \text{EB} \tag{3}$$

where length*(H)* and energy*(H)* are the schedule length and energy consumption of schedule H, respectively, and EB is the energy consumption constraint (also referred to as *Energy Budget*). The DAG scheduling problem is shown to be NP-complete in general [15]. Therefore, heuristic algorithms are needed to solve this problem in polynomial time.

Some notations are listed here for convenience:

- $\text{pred}(v_i)$ and $\text{succ}(v_i)$ 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 $[st, ft]$.

4. The Proposed EcoMapS Solution

Our proposed EcoMapS solution 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. The Initialization Phase algorithms aim to minimize schedule 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 run-time sensor failures. The scheduling algorithms are executed on cluster heads when applications are assigned to clusters. In case of

DOI: 10.1002/wcm

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, the main components of our proposed EcoMapS solution, namely, wireless channel modeling and Hyper-DAG extension, communication scheduling algorithm (CSA), extended CNPT algorithm [6] and Min-Min algorithm [7] (referred to as E-CNPT and E-MinMin, respectively), and Quick Recovery algorithm, are presented. During the Initialization Phase, either *E-CNPT* or *E-MinMin* is executed as the schedule search engine to find the optimal schedule. The original CNPT and Min-Min algorithms are designed for traditional parallel processing. To extend CNPT and Min-Min for WSNs, we developed a CSA based on the *wireless channel model* and the *Hyper-DAG representation* of applications. CSA is embedded in the execution of E-CNPT and E-MinMin. 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 or E-MinMin algorithm, which can be time consuming. The optimal schedule search with E-CNPT or E-MinMin will be executed only when the performance degrades to a certain threshold, a new application arrives, or the cluster head fails.

4.1. Wireless Channel Modeling and Hyper-DAG Extension

In single-hop clusters, there can be only one transmission at a given time. Similar to that in Reference [9], the wireless channel can be 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-network where all sensors only have connections with the virtual node $\mathcal{C}$. The communication latency between sensor nodes and $\mathcal{C}$ is considered zero since all wireless communications are accounted for by the tasks executed on $\mathcal{C}$. With the virtual node representation of $\mathcal{C}$, communication contention can be effectively avoided by serially scheduling communications on $\mathcal{C}$. Another important advantage of the channel model is its suitability to represent the broadcast nature of wireless communication. When a node in a single-hop cluster transmits information, it is potentially received by all nodes in the cluster. Wireless broadcasting can be leveraged to relay information generated by a task to its immediate successors in a single transmission rather than multiple transmissions. This approach reduces schedule lengths as well as communication energy consumption.

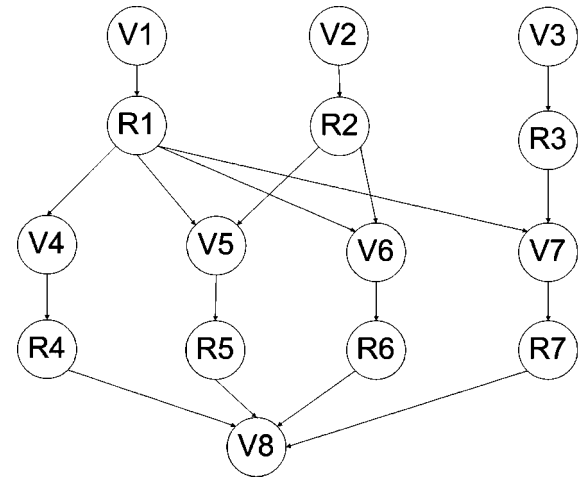


Fig 3. Hyper-DAG representation.

To implement this channel model, communication events between computation tasks should be explicitly represented in task graphs. Thus, we extend a DAG as follows: for a task v_i in a DAG, we replace the edges between v_i and its immediate successors with a *net* R_i. R_i represents the communication task to send the result of v_i to its immediate successors in the DAG. The weight of R_i equals to the result data volume of v_i. This extended DAG is a hypergraph and is referred to as *Hpyer-DAG*. With the Hyper-DAG representation, exclusive channel access constraints and broadcasting are incorporated into task dependency in a compact way. 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, $V = \{\text{Computation Tasks}\}$, and $R = \{\text{Communication Tasks}\}$. The example of converting the DAG in Figure 2 to a Hyper-DAG is shown in Figure 3.

In the Hyper-DAG scheduling problem, the *Dependency Constraint* is rephrased as follows: If a computation task v_j scheduled on node m_k depends on a communication task v_i on another node, a copy of 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.

4.2. CSA

Based on the Hyper-DAG and the channel model presented in Section 4.1, scheduling communication between single-hop neighbors is equivalent to first duplicating a communication task from the sender to $\mathcal{C}$, then from $\mathcal{C}$ to the receiver. If the requested communication task has been scheduled from the

DOI: 10.1002/wcm

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

Fig 4. Communication task scheduling algorithm.

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 receivers. The detailed description of the CSA is shown in Figure 4, where Steps 2–15 stand for originating a new communication from m_s to m_r, and Steps 18–21 represent reception of a broadcast data. Compared with originating a new communication, the broadcast method leads to energy saving of one data transmission for each additional data reception. With our CSA, one data transmission may reach multiple receivers, which enhances energy efficiency.

4.3. Task Mapping and Scheduling in Initialization Phase

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. These constraints together with the *Dependency Constraint* are represented as follows:

- A computation task can be assigned only on sensor nodes, that is, $\forall \gamma_i \in V : t_{v_i,\mathcal{C}} = \infty, c_{v_i,\mathcal{C}} = \infty$.
- A communication task can be assigned both on sensors and $\mathcal{C}$.
- If a communication task has its immediate predecessor and immediate successors assigned on the same node, it has zero execution length and energy cost.
- If a communication task is duplicated to or from $\mathcal{C}$, the corresponding data transmission or reception energy consumption is defined as Equation 1.
- If $v_i \in V$ and $\text{pred}(v_i) \neq \emptyset$, then $\text{pred}(v_i) \subset T(m(v_i))$ and $s_{v_i,m(v_i)} \geq \max f_{\text{pred}(v_i),m(v_i)}$.

During task mapping and scheduling, if a computation task depends on a communication task assigned on another sensor node, CSA will be executed to duplicate the absent communication task. With CSA 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 [15] and heuristic algorithms are needed to obtain practical solutions. In this section, E-CNPT and E-MinMin algorithms are presented for the Initialization Phase of EcoMapS with the objective of minimizing schedule lengths subject to energy constraints.

Before presenting E-CNPT and E-MinMin, 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 Reference [10], where only one sensor, that is the cluster head, in a cluster can execute high-level tasks. In EcoMapS, there can be more than one computing sensors to speed up execution. However, this approach generally consumes more energy with more computing sensors because of the increased volume of communication between the sensors. Thus, the increment of number of computing sensors must be bounded by energy consumption constraints. In our EcoMapS, E-CNPT and E-MinMin will iteratively search for the optimal schedule with different number of computing sensors subject to energy constraints.

DOI: 10.1002/wcm

4.3.1. E-CNPT

The list-scheduling CNPT algorithm [6] is extended and implemented in the Initialization Phase of EcoMapS, and is denoted as E-CNPT. The objective of E-CNPT is to minimize schedule lengths subject to energy consumption constraints. The strategy of E-CNPT is to assign the tasks along the most critical path to the nodes with the earliest execution start time (EEST). 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. Similar to CNPT, E-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 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:

Listing Stage. The Listing Stage of E-CNPT is similar to that of CNPT [6]. The Earliest Start Time EST(v_i) of task v_i is first calculated 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. The formulas to calculate EST and LST are as follows:

$$\begin{aligned} \mathrm{EST}(v_i) &= \max_{v_m \in \mathrm{pred}(v_i)} \{\mathrm{EST}(v_m) + t_m\}, \ \mathrm{LST}(v_i) \\ &= \min_{v_m \in \mathrm{succ}(v_i)} \{\mathrm{LST}(v_m)\} - t_i \end{aligned} \tag{4}$$

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 $v_i \in R$. Then, the Critical Nodes (CN) are pushed into stack S in the decreasing order of their LST. Here, a CN is a node 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 queue L. The Listing Phase ends when the stack is empty, and the task graph is sequentialized into L. It should be noted that the EST and LST are for evaluating the critical path of DAGs, and do not represent the actual execution start time of tasks.

Sensor Assignment Stage. The design objective of our algorithm is to minimize schedule lengths subject to energy consumption constraints. Different from CNPT, E-CNPT iteratively searches the schedule space with different number of computing sensors in the Sensor Assignment Stage. Among these schedules, the one with the minimum schedule length under the energy consumption constraint is chosen as the solution. If no schedule meets the energy constraint, the best effort is made by choosing the one with the minimum energy consumption. The detailed description of the E-CNPT algorithm is shown in Figure 5.

In E-CNPT, 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 q is the upper bound of the number of computing sensors that can be involved in the schedule. 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* embedded with CSA. The basic strategy of the algorithm is to assign tasks to the sensor with the minimum EEST. SingleCNPT(L,q) is described in Figure 6, where EAT(m_k) is the Earliest Available Time of node m_k, and EEST(v_i, m_k) is the EEST of v_i on sensor m_k. Different from EST, EEST

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 ← ∞ /*optimal schedule length*/
2. E_min ← ∞ /*min energy consumption*/
3. FOR q = 1 to p
4.   H = SingleCNPT(L,q)
6.   IF energy(H) < E_min
7.     E_min ← energy(H)
8.     H_min ← H /*eng(H_min) = min*/
9.   IF energy(H) ≤ EB & length(H) < L^o
10.    L^o ← length(H)
11.    H^o ← H /*optimal schedule*/
12. IF E_min ≤ EB
13.   Return H^o
14. ELSE
15.   Return H_min
```

Fig 5. E-CNPT algorithm.

DOI: 10.1002/wcm

Input: task queue L; number of computing sensors q
Output: Schedule of tasks in L
SingleCNPT Algorithm:

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** all computing sensors $\{m_k\}$
8. Calculate EEST(v_i, m_k):
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 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 min (EEST(v_i, m^o))
16. Schedule v_i on m^o: $s_{v_i,m^o} \leftarrow EEST(v_i, m^o)$

Fig 6. SingleCNPT algorithm.

represents the actual execution start time of a task if assigned on a sensor node.

4.3.2. E-MinMin

The Min-Min algorithm is reported of satisfying performance with relatively low complexity [7]. Thus, the Min-Min algorithm [7] is extended for the Initialization Phase of EcoMapS, and is referred to as the E-MinMin algorithm.

Similar to E-CNPT, E-MinMin also searches for the schedule with the optimal number of computing sensors that has the smallest schedule length subject to the energy consumption constraint. E-MinMin's schedule searching algorithm is the same as E-CNPT in Section 4.3.1 except that the input of E-MinMin is the Hyper-DAG instead of the task queue *L*, and the core of the searching algorithm is *SingleMinMin* instead of *SingleCNPT*.

We now introduce the procedure SingleMinMin-(Hyper-DAG,*q*) that schedules the tasks of the Hyper-DAG with q computing sensors. The core of *SingleMinMin* is the fitness function. For each task-node combination (*v*,*m*), the fitness function $fit(v, m, \alpha)$ 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 energy consumption cost for the time cost. In 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 Reference [7], the following notations are introduced first:

- MFT(*v*,*m*) is the maximum finish time of the tasks assigned prior to task *v*.
- $f_{v,m}$ is the scheduled finish time of *v* on *m*.
- 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*: $\text{NPT}(v, m) = \frac{f_{v,m}}{\text{MFT}(v,m)}$.
- NPE(*v*,*m*) is the normalized energy consumption of assigning *v* on *m*: $\text{NPE}(v, m) = \frac{\text{PE}(v,m)}{\text{EB}}$.

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

$$\text{fit}(v, m, \alpha) = \alpha \cdot \text{NPE}(v, m) + (1 - \alpha) \cdot \text{NPT}(v, m) \tag{5}$$

SingleMinMin is presented in Figure 7, where $\Delta\alpha$ is the α sampling step, 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.

4.4. Sensor Failure Handling with Quick Recovery Algorithm

In WSNs, sensors are prone to failures. In case of sensor failures, the schedules created by the E-CNPT and E-MinMin in the Initialization Phase may not be feasible solutions. In such cases, 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 from sensor failures. The E-CNPT or E-MinMin scheduling algorithms will not be executed unless the performance of the recovered schedule degrades to certain threshold.

The strategy of the quick recovery algorithm is to merge the tasks of the failing sensor onto the working sensor that has the most idle time. If there are more

DOI: 10.1002/wcm

Input: Hpyer-DAG; number of computing sensors: q
Output: Schedule H^o of tasks in Hyper-DAG
SingleMinMin Algorithm:

1. **FOR** $\alpha = 0$; $\alpha \leq 1.0$; $\alpha += \Delta\alpha$
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
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. Calculate $fit(v_i, m_k, \alpha)$
13. Find m_i^o: $fit(v_i, m_i^o, \alpha) = \min$
14. Find (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 unassigned mappable tasks
18. Among schedules with different α
19. **IF** $\exists H : energy(H) \leq EB$
20. Return H^o: $length(H) = \min$
21. **ELSE**
22. Return $H : energy(H) = \min$

Fig 7. SinglMin algorithm.

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 1. The quick recovery algorithm is shown in Figure 8, where 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 1. *The recovered scheme H^s still meets the energy consumption budget constraint, that is, if* energy(H^o) $\leq$ EB, *then* 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) +

Input: The failing sensor m_f, original sensor set SS, original schedule H^o
Output: Recovered schedule H^s
QuickRecovery Algorithm:

1. **IF** $\exists m_{idle} \in SS - \{m_f\} : T(m_{idle}) = \emptyset$
2. Reassign T(m_f) onto m_{idle}
3. **ELSE** /*no idle sensor*/
4. Find $m^o \in SS - \{m_f\}$: IR(m^o) = min
5. $t \leftarrow 0$, $\Delta t \leftarrow 0$ /*initialization*/
6. **FOR** $v_i \in S = T(m_f) \cup T(m^o)$
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** $\{m_f/m^o\}$ = {sender/receiver of v_i^c}
14. Remove v_i^c from $\mathcal{C}$
15. **ELSE IF** $pred(v_i) \in S$
16. Schedule v_i right after $pred(v_i)$
17. **IF** $succ(v_i) \not\subseteq S$ /*affect 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. $\Delta t \leftarrow max(\Delta t, f_{v_i,m^o} - s_{v_i^c,\mathcal{C}})$
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 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*

Fig 8. Quick recovery algorithm

DOI: 10.1002/wcm

commEng(H^o) = energy(H^o) $\leq$ EB holds, as well. ■

4.5. Computational 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. Thus, the Hyper-DAG is $T' = (V', E')$, where $|V'| = 2v$ and $|E'| = 2e$. The time complexity of EcoMapS with E-CNPT is analyzed as follows:

- Listing Stage of E-CNPT: Similar to CNPT [6], the complexity is $O(v + e)$.
- SingleCNPT: The communication tasks have complexity of $v \cdot O(1) = O(v)$, the entry-tasks have complexity of $f \cdot O(p) = O(fp)$, other non-entry computation tasks have complexity of $(v - f) \cdot O(p) \cdot O(e/v)$. Hence, the overall complexity of SingleMapSchedule is $O(v) + O(fp) + (v - f) \cdot O(p) \cdot O(e/v)$. 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.
- EcoMapS with E-CNPT: The SingleCNPT algorithm will be called $O(p)$ times. Thus, the complexity of the whole algorithm is $O(v + e) + O(p) \cdot O(v^2 p) = O(p^2 v^2)$ for the worst case.

The time complexity of EcoMapS with E-MinMin is analyzed as follows:

- SingleMinMin: The complexity of SingleMinMin is dominated by the loop starting from Step 5, which is executed $O(v)$ times. Similarly to SingleCNPT, the complexity of the loop starting from Step 6 has the complexity of $O(v) \cdot O(p) \cdot O(e/v) = O(pe)$. Thus, SingleMinMin has the complexity of $O(pv^3)$ for the worst case.
- EcoMapS with E-MinMin: similar to the analysis of E-CNPT, the complexity is $O(p) \cdot O(pv^3)/\Delta\alpha = O(p^2 v^3/\Delta\alpha)$ for the worst case.

From the analysis above, the complexity of the EcoMapS with E-MinMin is higher than that of the EcoMapS with E-CNPT with the order of v for a fixed value of $\Delta\alpha$.

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 $O(2v - 1) = O(v)$.

5. Simulation Results

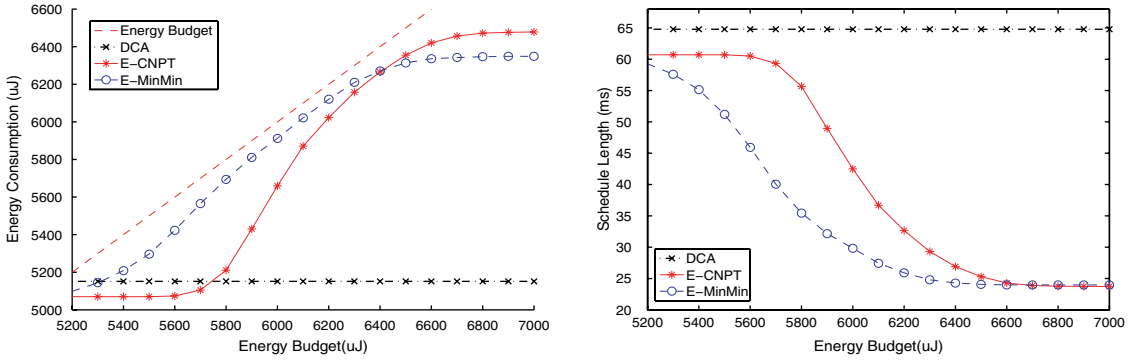
The performance of our EcoMapS solution with E-CNPT and E-MinMin is investigated through simulations. The performance of an extended version of DCA [10] is evaluated as a benchmark. DCA is extended with our proposed CSA to deliver the intermediate results of entry-tasks to the cluster head for further processing. Simulations are run on arbitrary applications with randomly generated DAGs. Our simulations study the effect of the following factors: (1) energy consumption constraints, (2) number of tasks in applications, (3) the inter-task dependency. We further evaluate the schedule energy consumption balance of EcoMaps, investigate the performance of the Quick Recovery Algorithm, and compare the heuristic execution time of E-CNPT and E-MinMin. In these simulations, we observe energy consumption and schedule lengths unless otherwise stated. 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. The presented simulation results correspond to the average of 500 independent runs.

5.1. Simulation Parameters

In our simulation study, the bandwidth of the channel is set to 1 Mb/s and the transmission range $r = 10$ meters. We assume that there are 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 1, 2 are in coherence with [10,13,14] as follows: $E_{\text{elec}} = 50$ nJ/b, $\varepsilon_{\text{amp}} = 10$ 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. Our simulation shows that when $\Delta\alpha = 0.1$, the E-MinMin algorithm provides satisfying performance with reasonable heuristic execution time. Thus, $\Delta\alpha$ is set to be 0.1 for our simulation studies.

To evaluate EcoMapS performance for arbitrary applications, simulations are run on randomly generated DAGs 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 immediate predecessors, computation load (in units of kilo-clock-cycle, KCC), and resulting data volume (in units of bit) of a task are uniformly distributed over [1, maxPred], [300 K CC $\pm$10%], and [800 bits $\pm$10%], respectively.

DOI: 10.1002/wcm



(a) (b)

Fig 9. Effect of the energy consumption constraints; (a) Energy consumption; (b) Schedule length.

5.2. Effect of the Energy Consumption Constraints

The effect of the energy consumption constraints is evaluated with randomly generated DAGs with numTask = 25, numEntry = 6, and maxPred = 3. As shown in Figure 9, both EcoMapS algorithms have better capability to adjust their schedules according to energy budget than DCA. When the energy budget is small, EcoMapS algorithms converge to use one sensor for computation, which is the default behavior for DCA. Instead of sending all sensed data to cluster heads, the EcoMapS algorithms choose one of the sensing sensor for computation, which saves energy and shortens schedule lengths. As energy budget increases, the EcoMapS algorithms have 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 higher availability of energy resources. Compared with DCA, EcoMapS can lead up to 67% schedule length improvement for this set of simulations.

Regarding the comparison of the EcoMapS algorithms themselves, both E-CNPT and E-MinMin tend to use one computing sensor with small energy budget, which leads to equal schedule lengths and energy consumptions. When the energy budget is sufficiently large, E-CNPT has a slightly shorter schedule length than E-MinMin because of its better perspective of global optimization: The listing stage of E-CNPT enqueues tasks according to the critical path of Hyper-DAG, while E-MinMin just locally calculates the cost of assigning a task. However, this improvement comes at a higher energy consumption cost, as shown in Figure 9a. For the scenarios with intermediate energy budgets, E-MinMin outperforms E-CNPT up to 39% in terms of schedule lengths (Figure 9b). This advantage of E-MinMin stems from its fitness function. Different from E-CNPT, which just takes time cost into account when assigning tasks with the fixed number of computing sensors, the fitness function of E-MinMin considers time cost as well as energy consumption. Thus, E-MinMin is more likely to find a feasible schedule meeting energy constraints with a larger number of computing sensors than E-CNPT, which leads to shorter schedule lengths.

5.3. 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). As shown in Figure 10, energy consumption and schedule lengths are dominated by the number of tasks. When the number of tasks increases, the energy consumption and schedule length of DCA increase proportionally. The EcoMapS algorithms on the other hand adapt themselves to the increasing energy budget. For the extreme scenarios with small and large energy budgets, the schedule lengths and energy consumption of the EcoMapS algorithms increase in proportion to the number of tasks. For the intermediate scenarios, the EcoMapS algorithms adapt their schedule lengths and energy consumptions according to the available energy budget when the number of tasks increases. For all three scenarios, the energy consumption of E-MinMin follows energy budgets closer than E-CNPT, and the schedule length of E-MinMin is shorter than E-CNPT for the scenarios with intermediate energy budgets.

DOI: 10.1002/wcm

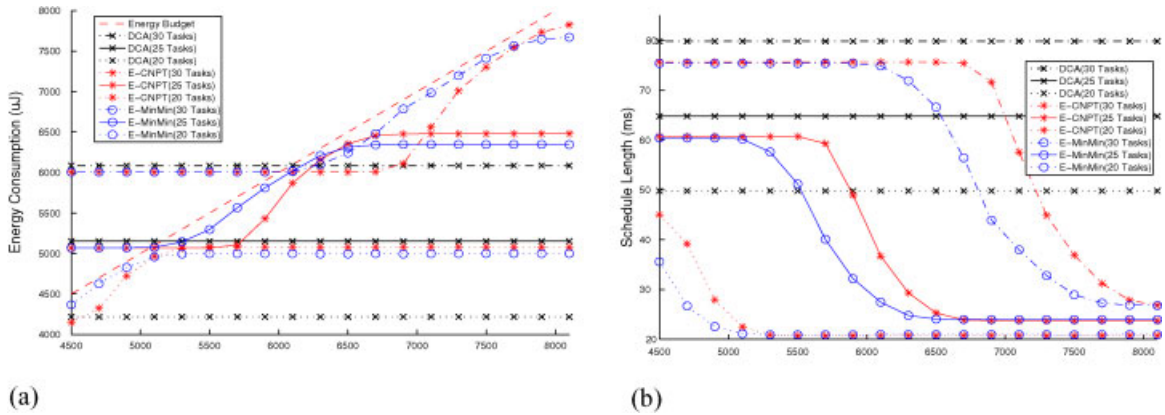


Fig 10. Effect of Number of Tasks; (a) Energy consumption; (b) Schedule length.

5.4. 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. As shown in Figure 11, the inter-task dependency has almost no effect on 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. Regarding the EcoMapS algorithms, increasing the in/out degree of DAGs does not introduce new communication task in Hyper-DAGs, but increases the dependency between a communication task and its immediate successors. Greater dependency degree may lead 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. Thus, when the energy budget is sufficiently larger, the energy consumption of the EcoMapS algorithms increases and the schedule lengths decrease. When the energy budget is relatively tight, both of the EcoMapS algorithms use less computing sensors to meet energy constrains when the inter-task dependency increases, which decreases energy consumptions and increases schedule lengths. As we can see from Figure 11b, although the performances of the EcoMapS algorithms degrade with higher inter-task dependency, the EcoMapS algorithms still outperform DCA with respect to schedule lengths subject to energy consumption constraints.

5.5. Evaluation of the Energy Consumption Balance

In this section, the energy consumption balance of the proposed EcoMapS algorithms are evaluated and compared to the DCA algorithm. The random DAGs considered in the simulations have the parameters of numTask = 25, numEntry = 6, and maxPred = 3. The observed metrics are the *Fairness Index (FI)* and the

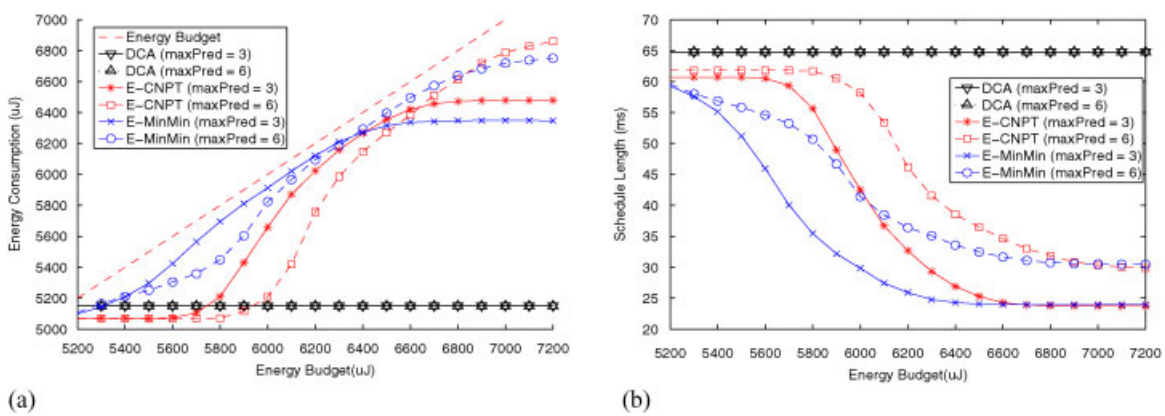


Fig 11. Effect of inter-task dependency; (a) Energy consumption; (b) Schedule length.

DOI: 10.1002/wcm

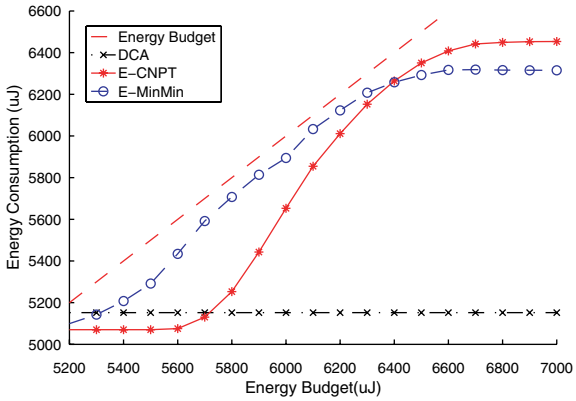


Fig 12. Overall energy consumption.

Maximum of Sensor' Energy Consumption in addition to the energy consumption of all sensors. Here, the FI is a variation of Jain's FI [16], and is defined as $\mathrm{FI} = \frac{(\sum_{k=1}^{n} E_k)^2}{n \sum_{k=1}^{n} E_k^2}$, where E_k is the energy consumption of sensor m_k, and n is the number of active sensors. The 'active sensors' are the sensors that execute either entry-tasks or non-entry-tasks. FI varies in [0,1], and the closer of FI to 1 , the better the energy consumption balance of the schedule.

As shown in Figure 12, when the energy budget is small, the EcoMapS algorithms tend to utilize a small number of computing sensors to reserve energy. Thus, most computation as well as energy consumption are burdened on these sensors (Figure 13b), which leads to relatively inferior energy consumption balance (Figure 13a). When the energy budget increases, more sensors can be involved in the application execution with the increased overall energy consumption increases due to the larger communication volume. However, the maximum of each sensor's energy consumption decreases (Figure 13b) and the energy consumption balance improves (Figure 13a) because of the distributed computation load among sensors. Compared to E-CNPT, the energy consumption of E-MinMin is more balanced for the scenarios with intermediate energy budgets. On the other hand, DCA always overloads the cluster head with most computation tasks regardless of energy availability, which causes poor energy consumption balance for all scenarios.

5.6. Performance of the Quick Recovery Algorithm

The Quick Recovery Algorithm is evaluated in this section. Since the recovery mechanism with idle sensors 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 using the quick recovery algorithm presented in Section 4.4. From Figure 14a, it can be observed 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 1, task merging leads to less energy consumptions at the cost of longer schedule lengths according to Figure 14b.

5.7. Comparison of the Heuristic Execution Time

The relative execution time of E-MinMin over E-CNPT is tested with randomly generated DAGs of different number of tasks (25, 30, 35, 40, 45, and 50) with numEntry = 6 and maxPred = 3. According to our simulation results, E-CNPT is slightly more than 50 times faster than E-MinMin for tested scenarios. When the number of tasks increases, the

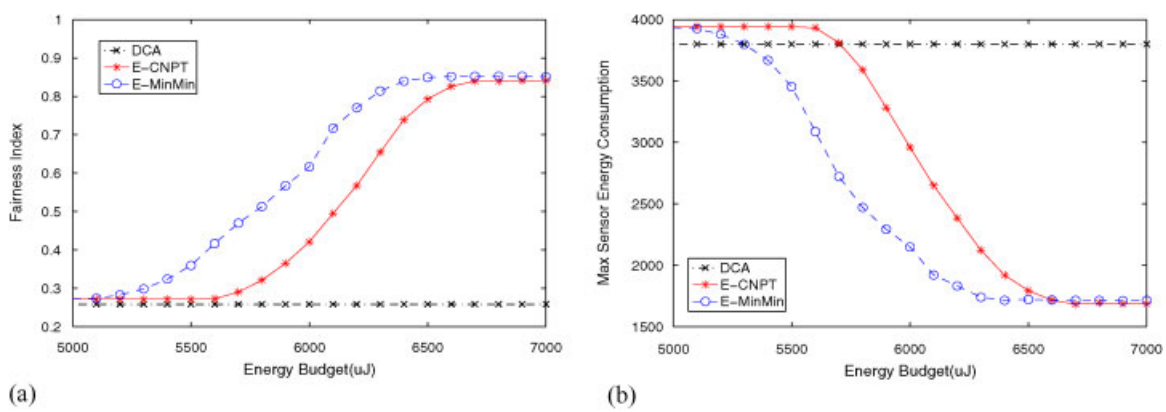


Fig 13. Energy consumption balance; (a) fairness index; (b) maximum energy consumption per node.

DOI: 10.1002/wcm

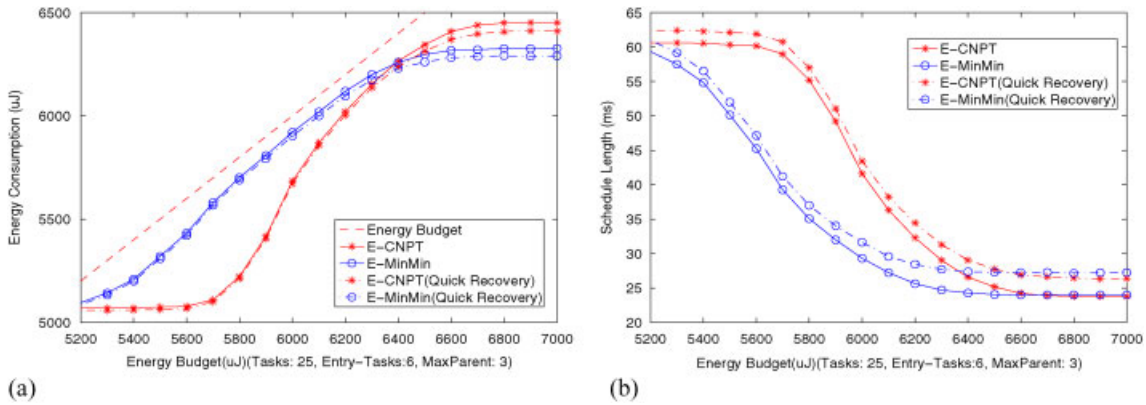


Fig 14. Performance of quick recovery; (a) Energy consumption; (b) Schedule length.

speed difference between E-CNPT and E-MinMin also slightly increases.

As shown in the previous sections, the performances of E-CNPT and E-MinMin are similar when the energy budget is small or sufficiently large. For such scenarios, E-CNPT is more preferable with shorter execution time. For the scenarios with medium energy budgets, E-MinMin generally provides shorter schedules with better energy consumption balance than E-CNPT. However, taking the heuristic execution time into account, the trade-off between the schedule length and the heuristic execution time should be considered. Both of E-CNPT and E-MinMin are executed in the Initialization Phase of EcoMapS, and schedules must be regenerated when new applications arrive. For WSNs without frequent application updates, the scheduling overhead is negligible and E-MinMin is preferred because of its shorter schedule lengths. For a WSN that updates its applications more frequently, E-CNPT can be favored over E-MinMin due to E-CNPT's shorter schedule computation time.

6. Conclusions

Parallel processing among sensors is a promising solution to provide the computation capacity required by in-network processing. In this paper, we present a task mapping and scheduling solution, EcoMapS, in single-hop clustered homogeneous WSN clusters. EcoMapS aims to minimize schedule lengths of applications under energy consumption constraints. Incorporating the channel model and communication scheduling algorithm, EcoMapS simultaneously schedules communication and computation tasks. A quick recovery algorithm is also proposed to handle sensor failures. Simulations show that EcoMapS provides superior performance in terms of schedule length and energy consumption balance compared to the DCA algorithm. The E-MinMin-based EcoMapS algorithm outperforms the E-CNPT-based EcoMapS algorithm with respect to schedule lengths and energy consumption balance but has a larger computation overhead. Thus, the E-MinMin-based EcoMapS algorithm is suitable for WSN applications that do not change frequently, while the E-CNPT-based EcoMapS is preferable where application updates occur more frequently. Our future work includes extending our wireless channel model to multi-hop clusters, addressing communication failures, and extending our solutions to heterogeneous WSNs.

References

1. Akyidiz IF, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey. *Computer Networks (Elsevier) Journal* 2002; **38**(4): 393–422.
2. Vercauteren T, Guo D, Wang X. Joint multiple target tracking and classification in collaborative sensor networks. *IEEE Journal on Selected Areas in Communication* 2005; **23**(4): 714–723.
3. Feng W-C, Kaiser E, Feng WC, Baillif ML. Panoptes: scalable low-power video sensor networking technologies. *ACM Transactions on Multimedia Computing, Communications, and Applications* 2005; **1**(2): 151–167.
4. Valera M, Velastin SA. Intelligent distributed surveillance systems: a review. *IEEE Proceedings on Vision, Image and Signal Processing* 2005; **152**(2): 192–204.
5. Dogan A, Özgüner F. Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogenous computing. *IEEE Transactions on Parallel and Distributed Systems* 2002; **13**(3): 308–323.
6. Hagras T, Janecek J. A high performance, low complexity algorithm for compile-time job scheduling in homogeneous computing environments. In *Proceeding of International Conference on Parallel Processing Workshops (ICPPW'03)* 2003; pp.149–155.
7. Shivle S, Castain R, Siegel HJ, *et al.* Static mapping of subtasks in a heterogeneous ad hoc grid environment. In *Proceedings of Parallel and Distributed Processing Symposium* 2004.
8. Yu Y, Prasanna VK. Energy-balanced task allocation for collaborative processing in wireless sensor networks. *ACM/Kluwer*

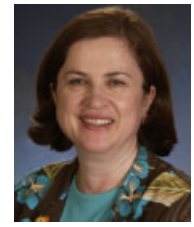
DOI: 10.1002/wcm

Journal of Mobile Networks and Applications 2005; **10** (1–2): 115–131.
9. Giannecchini S, Caccamo M, Shih C-S. Collaborative resource allocation in wireless sensor networks. In *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS'04)* 2004; pp.35–44.
10. Wang A, Chandrakasan A. Energy-efficient DSPs for wireless sensor networks. *IEEE Signal Processing Magazine* 2002; 68–78.
11. So J, Vaidya NH. Multi-channel MAC for ad hoc networks: handling multi-channel hidden terminals using a single transceiver. In *Proceedings of ACM MobiHoc'04* 2004; pp. 222–233.
12. Hong Y-W, Scaglione A. Energy-efficient broadcasting with cooperative transmissions in wireless sensor networks. *IEEE Transactions on Wireless Communications* 2006; **5**(10): 2844–2855.
13. Heinzelman WB, Chandrakasan AP, Balakrishnan H. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications* 2002; **1**(4): 660–670.
14. Shih E, Cho S, Ickes N, *et al.* Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *Proceedings of ACM MobiCom'01* 2001; pp. 272–286.
15. Garey M, Johnson D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co.: New York, NY, 1979.
16. Jain R, Chiu D-M, Hawe W. A quantitative measure of fairness and discrimination for resource allocation in shared conputer systems. Tech. Rep. TR-301, DEC Research Report, September 1984.

Authors' Biographies

Yuan Tian received his B.S. and M.S. degrees in Information Engineering and Communication and Information System from Zhejiang University, Hangzhou, China, in 1998 and 2001, respectively. He recently received his Ph.D. degree in Electrical and Computer Engineering from the Ohio State University, Columbus, OH, in 2006. He currently works in the Wireless Technology Group of the Bosch Research and Technology Center North America, Palo Alto, CA. Dr Tian's current research interests include wireless sensor networks, wireless network architecture, and network resource allocation and optimization.

Eylem Ekici has received his B.S. and M.S. degrees in Computer Engineering from Bogazici University, Istanbul, Turkey, in 1997 and 1998, respectively. He received his Ph.D. degree in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, GA, in 2002. Currently, he is an Assistant Professor in the Department of Electrical and Computer Engineering of the Ohio State University, Columbus, OH. Dr Ekici's current research interests include wireless sensor networks, vehicular communication systems, next generation wireless systems, and space-based networks, with a focus on routing and medium access control protocols, resource management, and analysis of network architectures and protocols.



Fusun Ozguner received the M.S. degree in Electrical Engineering from the Istanbul Technical University in 1972, and the Ph.D. degree in Electrical Engineering from the University of Illinois, Urbana-Champaign, in 1975. She worked at the I.B.M. T.J. Watson Research Center with the Design Automation group for 1 year and joined the faculty at the Department of Electrical Engineering, Istanbul Technical University in 1976. Since January 1981 she has been with The Ohio State University, where she is presently a Professor of Electrical and Computer Engineering. Her current research interests are parallel and fault-tolerant architectures, heterogeneous distributed computing, reconfiguration and communication in parallel architectures, real-time parallel computing and communication, and wireless networks. Dr Ozguner has served as an Associate Editor of the *IEEE Transactions on Computers* and on program committees of several international conferences.

DOI: 10.1002/wcm