Modeling Optimal Dynamic Scheduling for Energy-aware Workload Distribution in Wireless Sensor Networks

Wanli Yu, Yanqiu Huang, Alberto Garcia-Ortiz
Institute of Electrodynamics and Microelectronics, University of Bremen, 28359 Bremen, Germany
Email:{wyu, huang, agarcia}@item.uni-bremen.de

***Abstract*—Energy-aware workload distribution becomes crucial for extending the lifetime of wireless sensor networks (WSNs) in complex applications as those in Internet-of-Things or in-network DSP processing scenarios. Today static workload schedules are well understood, while dynamic schedules (i.e., with multiple partitions) remain unexplored. This paper models the dynamic scheduling by considering both the communication and computation energy consumption. It formulates a series of (integer) linear programming problems to characterize the optimal scheduling strategies. Surprisingly, even 2-partition scheduling can provide the maximum gains. Besides the interest to evaluate the optimality of on-line heuristics for dynamic scheduling, the reported off-line strategies can be immediately applied to WSN applications.**

I. Introduction

Energy-aware workload distribution is one of the most efficient approaches to reduce energy consumption [1]. It has been intensively used in grids and multiprocessors, etc.; however, it is rarely studied in wireless sensor networks (WSNs), especially considering dynamic schedules. The reason is that in the traditional applications of WSNs, the workloads are very simple and wireless communication is usually the most energy intensive process [4]. However, as the applications of WSNs become more complex, e.g. in-network DSP processing and Internet-of-Things, it needs to efficiently distribute the workloads by considering both the computation and communication energy consumption.

A few works manage to balance the workload by providing the static distribution schedules with one constant partition cut [5] [2]. The achievements are expected to be improved by the dynamic scheduling with multiple partition cuts, which, however, have not been investigated in WSNs. To the best of our knowledge, this is the first paper that models the dynamic workload distribution scheduling and provides the performance comparisons with respect to the static ones.

II. Modeling the Workload Distribution and Energy Consumption for Cluster-based WSNs

This section models the workload distribution and the energy consumption in the cluster-based WSNs, where the whole workload is completed through the cooperation of the slave and the master nodes.

A. Modeling Workload Distribution

A WSN workload can be represented by a synchronous dataflow (SDF) graph [2] [3], $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, as shown in Fig. 1. Each actor $v \in \boldsymbol{V}$ is a computational module for executing task; each edge $e \in \boldsymbol{E}$ represents a buffer for actors to store and fetch data (tokens). The tokens consumed and generated by an actor, $k_c(v)$ and $k_g(v)$, are:

$$k_c(v) = \sum_{e \in in(v)} k_o(e), \quad k_g(v) = \sum_{e \in out(v)} k_i(e) \qquad (1)$$

where $in(v)$ and $out(v)$ are its input and output edges; $k_o(e)$ and $k_i(e)$ are the output and input tokens on edge e. For each edge, the total amount of its input tokens generated by the source actor, $src(e)$, equals its output tokens consumed by the sink actor, $snk(e)$, i.e., $q(src(e)) \cdot k_i(e) = q(snk(e)) \cdot k_o(e)$, where $q(\cdot)$ is the number of execution times of the actor.

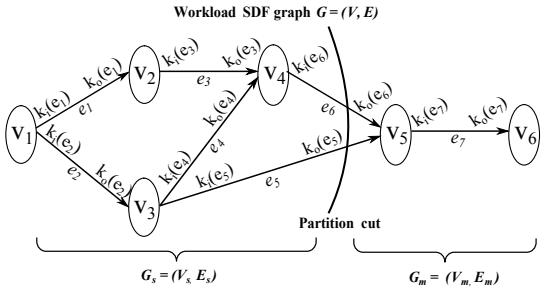


Fig. 1: An example of the synchronous dataflow (SDF) graph

Distributing the workload among the slave and master nodes is equivalent to divide the modeled SDF graph, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, into two subgraphs, $\boldsymbol{G_s} = (\boldsymbol{V_s}, \boldsymbol{E_s})$ and $\boldsymbol{G_m} = (\boldsymbol{V_m}, \boldsymbol{E_m})$, with a partition cut, $\boldsymbol{X} = [x(v_1), \cdots, x(v_l), \cdots, x(v_L)]^T$. L is the number of actors in the SDF graph; $x(v_l)$ is a boolean value defined as: $x(v_l) = 0, \forall v_l \in \boldsymbol{G_s}$; $x(v_l) = 1, \forall v_l \in \boldsymbol{G_m}$.

To guarantee that a) the workload is completed by both the slave and master nodes, and b) the data, k_d, is transmitted from the slave to the master, **it has to satisfy:** a) $1 \leq \mathbb{1}_{1\times L} \cdot \boldsymbol{X} \leq L - 1$ and

b) $\mathbb{B}^T \cdot \boldsymbol{X} \leq \mathbb{0}$, where $\mathbb{1}_{1\times L}$ is a $1 \times L$ all one vector; $\mathbb{B}$ is the incidence matrix of the SDF graph; $\mathbb{0}$ is an all zero vector. Let $\boldsymbol{K_v} = [k_n(v_1), \cdots, k_n(v_l), \cdots k_n(v_L)]$ denote the vector of the net consumed tokens of each actor, where $k_n(v_l) = q(v_l) \cdot \big(k_c(v_l) - k_g(v_l)\big)$, the transmitted data, k_d, can be formulated as:

$$k_d = \boldsymbol{K_v} \cdot \boldsymbol{X} = \boldsymbol{K_v} \cdot (\boldsymbol{X} - \mathbb{1}^T_{1\times L}) \qquad (2)$$

Notice that $\boldsymbol{K_v} \cdot \mathbb{1}^T_{1\times L} = 0$ in the SDF graph.

B. Modeling Energy Consumption

This subsection models the energy consumption including computation and communication cost [1].

[1]The sleeping energy cost is not modeled here, which is typically very small. The interested reader can refer [2] for a more detailed formulation.

The total computation energy consumption for slave i and master on a given partition, $\boldsymbol{X_i}$, in one scheduling period are:

$$\begin{aligned} E_{cp\_si}(\boldsymbol{X_i}) =& \boldsymbol{E_{cp\_si}} \cdot \left(\mathbb{1}_{1\times L}^T - \boldsymbol{X_i}\right) \\ E_{cp\_m}(\boldsymbol{X_i}) =& \boldsymbol{E_{cp\_m}} \cdot \boldsymbol{X_i} \end{aligned} \tag{3}$$

where $\boldsymbol{E_{cp\_si}} = \left[e_{cp\_si}(v_1), \cdots, e_{cp\_si}(v_l), \cdots, e_{cp\_si}(v_L)\right]$ and $\boldsymbol{E_{cp\_m}} = \left[e_{cp\_m}(v_1), \cdots, e_{cp\_m}(v_l), \cdots, e_{cp\_m}(v_L)\right]$ are the vectors of the computation energy cost of each actor when it is executed in slave i and the master node, respectively. The calculation of $e_{cp\_si}(v)$ and $e_{cp\_m}(v)$ can be found in [5].

The communication energy cost of slave node i and the master node for transmitting and receiving k_d bits data are:

$$\begin{aligned} E_{cm\_si}(\boldsymbol{X_i}) &= e_{o\_si} + e_{tx\_si} \cdot k_d \\ &= e_{o\_si} + e_{tx\_si} \cdot \boldsymbol{K_v} \cdot (\boldsymbol{X_i} - \mathbb{1}_{1\times L}^T) \\ E_{cm\_m}(\boldsymbol{X_i}) &= e_{o\_m} + e_{rx} \cdot k_d \\ &= e_{o\_m} + e_{rx} \cdot \boldsymbol{K_v} \cdot \boldsymbol{X_i} \end{aligned} \tag{4}$$

where e_{o_si} and e_{o_m} are the energy of slave i and the master spent on the overhead activities [2]; e_{tx_si} and e_{rx} are the corresponding energy cost for transmitting and receiving one bit data, respectively.

Since the master node is in charge of all the n slave nodes, it has to iterate n times to receive and process the data. Combing Eq. (3) and (4), the energy consumption of slave i and the master node in one scheduling period can be expressed as:

$$\begin{aligned} E_{si}(\boldsymbol{X_i}) &= E_{cp\_si}(\boldsymbol{X_i}) + E_{cm\_si}(\boldsymbol{X_i}) \\ E_m(\boldsymbol{X_1}, \cdots, \boldsymbol{X_n}) &= \sum_{i=1}^{n} E_{cp\_m}(\boldsymbol{X_i}) + E_{cm\_m}(\boldsymbol{X_i}) \end{aligned} \tag{5}$$

III. Energy-aware Workload Distribution Schedules for Maximizing The Network Lifetime

For generality, we consider an asymmetrical network, where each slave node needs an individual partition solution. The network lifetime is defined as the time when the first node runs out of energy, which is a very popular definition [2] [6].

A. Static Scheduling for Workload Distribution

The policy of the static scheduling is to use an individual partition cut to distribute the workload for each slave and master nodes during all time. The nodes in each scheduling period consume fixed energy. The problem of maximizing the network lifetime T_{net} is equivalent to minimize its reciprocal $1/T_{net}$. It can be formulated as a binary integer linear programming (BIP) problem:

$$\begin{aligned} &\arg\min_{\boldsymbol{X_i}} \max\left\{\frac{E_m(\boldsymbol{X_1}, \cdots, \boldsymbol{X_n})}{B_m}, \frac{E_{si}(\boldsymbol{X_i})}{B_{si}}\right\}, i = 1, \cdots, n \\ &\text{subject to :} \\ &\quad 1 \leq \mathbb{1}_{1\times L} \cdot \boldsymbol{X_i} \leq L - 1 \quad \& \quad \mathbb{B}^T \cdot \boldsymbol{X_i} \leqq \mathbb{0} \end{aligned} \tag{6}$$

By solving this BIP problem the static schedule can maximize the network lifetime with one constant partition.

B. Dynamic Scheduling for Workload Distribution

In the real scenario, the workload cannot be distributed equally with only one partition cut. An example is shown in Fig. 2, executing actor v_2 costs much more energy than v_1 and v_3. Assuming C_2 is the partition calculated by the static scheduling, the slave node will die soon, while the master still has plenty of energy. If both cuts are used, the energy consumption of slave and master can be more balanced. This motivates us to study two concrete multi-partition schedules.

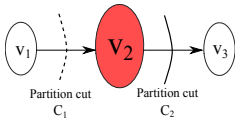


Fig. 2: A simple example of the schedule with 2 partitions

1) Cyclic scheduling: The policy of cyclic scheduling selects a given set of partition cuts, they are iteratively used one after another. The optimal solution can be calculated by copying the original SDF graph $nc-1$ times (nc is the number of the expected partition cuts) to obtain an extended graph, then using Eq. (6) to calculate the big partition $\boldsymbol{X_i}$ which is a $nc{\cdot}L{\times}1$ vector that consists of nc partitions $\boldsymbol{X_{i\_1}}, \cdots, \boldsymbol{X_{i\_nc}}$.

As nc increases, some partition cuts may be repeated due to the limited amount of possible partition cuts in a SDF graph. The cyclic scheduling is actually selecting partition cuts with the corresponding weights. When nc approaches infinity, it can find the best solution to achieve the global maximum network lifetime. But solving the BIP problem for larger nc also becomes very computational expensive. We address this issue in the following section.

2) Weighted scheduling: A weighted scheduling is proposed to reduce the computation complexity of the cyclic scheduling. It provides the ideal partition cuts with the corresponding weights (probabilities), through calculating the probability of each actor that belongs to the slave or the master.

Let $E_{si}^j(\boldsymbol{X_i^j})$ denote the energy cost of slave node i when exploiting the partition $\boldsymbol{X_i^j}$, and $E_m^j(\boldsymbol{X_1^j}, \cdots, \boldsymbol{X_n^j})$ denote the cost of the master node in the jth scheduling round. Assuming that the network collapses after J rounds, according to Eq. (5), the average energy cost of the slave and the master nodes in one scheduling round can be formulated as:

$$\begin{aligned} \overline{E}_{si}(\boldsymbol{\Gamma_i}) &= \frac{1}{J} \cdot \sum_{j=1}^{J} E_{si}^j(\boldsymbol{X_i^j}) \\ &= e_{o\_si} + \left(\boldsymbol{E_{cp\_si}} - e_{tx\_si} \cdot \boldsymbol{K_v}\right) \cdot \left(\mathbb{1}_{1\times L}^T - \boldsymbol{\Gamma_i}\right) \\ \overline{E}_m(\boldsymbol{\Gamma_i}, \cdots, \boldsymbol{\Gamma_n}) &= \frac{1}{J} \cdot \sum_{j=1}^{J} E_m^j(\boldsymbol{X_1^j}, \cdots, \boldsymbol{X_n^j}) \\ &= \sum_{i=1}^{n} e_{o\_m} + \left(\boldsymbol{E_{cp\_m}} + e_{rx} \cdot \boldsymbol{K_v}\right) \cdot \boldsymbol{\Gamma_i} \end{aligned} \tag{7}$$

where $\boldsymbol{\Gamma_i} = \frac{1}{J}\sum_{j=1}^{J} \boldsymbol{X_i^j} = [\gamma_i(v_1), \cdots \gamma_i(v_l), \cdots \gamma_i(v_L)]^T$. Each element $\gamma_i(v_l) = \frac{1}{J} \cdot \sum_{j=1}^{J} x_i^j(v_l)$, satisfying $0 \leq \gamma_i(v_l) \leq 1$, is the probability that how often the actor v_l is executed by the master. Consequently, maximizing the network lifetime can be formulated as the following linear programming (LP) problem:

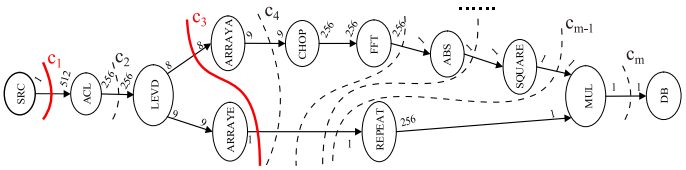
$$\begin{aligned} &\arg\min_{\boldsymbol{\Gamma_i}} max\left\{\frac{\overline{E}_m(\boldsymbol{\Gamma_i}, \cdots, \boldsymbol{\Gamma_n})}{B_m}, \frac{\overline{E}_{si}(\boldsymbol{\Gamma_i})}{B_{si}}\right\}, i = 1, \cdots, n \\ &\text{subject to:} \\ &\quad 1 \leq \mathbb{1}_{1\times L} \cdot \boldsymbol{\Gamma_i} \leq L - 1 \quad \& \quad \mathbb{B}^T \cdot \boldsymbol{\Gamma_i} \leqq \mathbb{0} \end{aligned} \tag{8}$$

This LP problem can be solved with the help of the optimization tools. After obtaining $\boldsymbol{\Gamma_i}$, it is straightforward to calculate the corresponding partition cuts and their weights. An example is illustrated in the simulation.

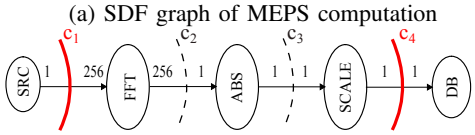
IV. Simulation Results and Discussion

In this section, we present the initial simulation results to estimate the potential of the dynamic schedules by comparing with the static scheduling.

For simplicity, we focus on one cluster. It can be easily extended to multiple clusters by iterating the schedules. The topologies of the networks are randomly generated and the battery energy of each node varies from 10 J to 50 J. Two typical computation applications, MEPS and spectrum are used, and the energy related parameters can be found in [2] and [5]. The corresponding SDF graphs are shown in Fig. 3a and Fig. 3b. Each simulation is repeated 100 times with random networks and the results correspond to the average values.



(a) SDF graph of MEPS computation



(b) SDF graph of spectrum computation

Fig. 3: The SDF graphs of the applcations of MEPS and spectrum computations, and the related possible partition cuts.

Fig. 3 depicts the increase of network lifetime when exploiting scheduling policies w.r.t. no scheduling, in which each slave node just executes the first actor. Obviously, both static and dynamic schedules can drastically extend the network lifetime. As n increases, the gains become more significant. Taking MEPS computation for example: the static schedules extend the network lifetime from 46% to 447% when n increases from 1 to 10. The reason is that without scheduling the master dies soon due to the increasing workload.

Moreover, the improvements of the dynamic schedules are even larger compared to the static one. The cyclic scheduling extends 2 and 4 times of the improvement of the static scheduling when using 2 cuts as shown in Fig. 4a and Fig. 4b, respectively. As the number of the partitions nc increases, the gains gradually rise and finally saturate. When $n = 5$ as shown in Fig. 4d, the maximum gain provided by the weighted schedule is 336%, which is 22.12% longer than the static solution.

Further on, one remarkable result is that there are always only two different cuts in the optimal partition solutions regardless of applications, topologies and network sizes. For instance, in one of the generated network with 10 slave nodes, the Γ_i for spectrum computation returned by the weighted scheduling is $\mathbf{\Gamma_i} = [0 \quad 0.1702 \quad 0.1702 \quad 0.1702 \quad 1] =$. It iss equals $0.1702 \cdot [0 \quad 1 \quad 1 \quad 1 \quad 1] + 0.8298 \cdot [0 \quad 0 \quad 0 \quad 0 \quad 1]$, which corresponds to the partition cuts C_1 and C_4 with weights 17.02% and 82.98% as shown in Fig. 3b. Consistently, the partition cuts C_1 and C_3 with weights 36.26% and 63.74% are provided for the MEPS application as shown in Fig. 3a.

V. Conclusions

This work models the dynamic workload schedules in WSNs. It formulates a series of (integer) linear programming problems to characterize the optimal scheduling strategies. The initial simulation results show that the dynamic schedules extend the network lifetime longer than the static one. Remarkably, the maximum gains can be achieved by using only 2 partition cuts regardless of applications, topologies and network sizes.

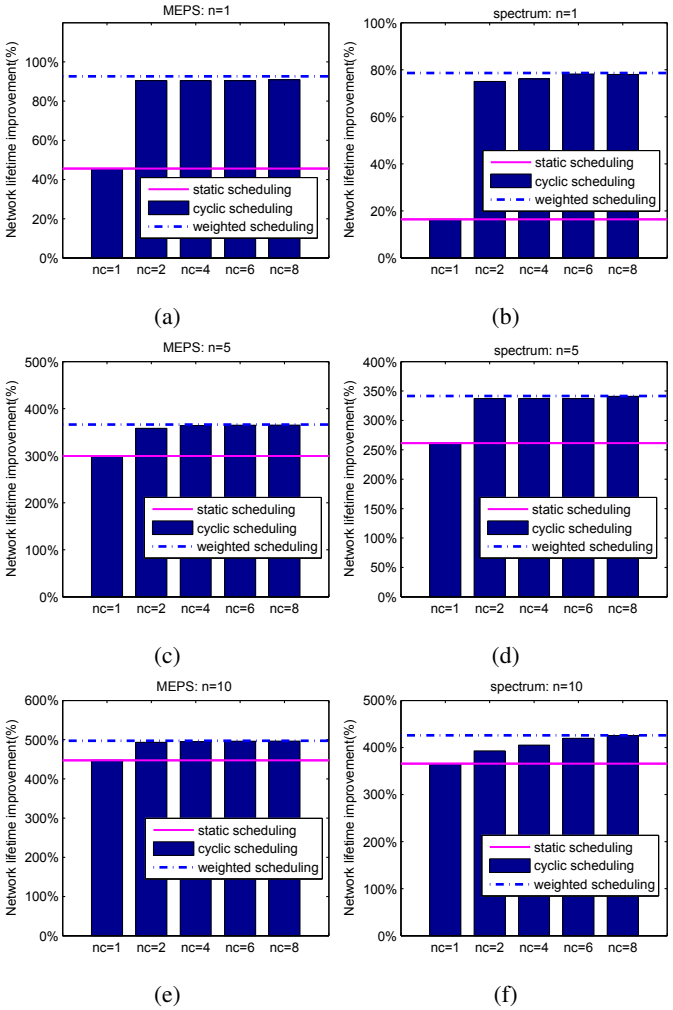


Fig. 4: Comparisons of the performance on extending network lifetime when exploiting static and dynamic schedules with respect to no workload distribution scheduling.

References

[1] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7(2), pp. 279–301, 1989.

[2] Y. Huang, et.al. Accurate energy-aware workload distribution for wireless sensor networks using a detailed communication energy cost model. *Journal of Low Power Electronics*, 10(2), pp. 183–193, 2014.

[3] D.-I. Ko, et.al. Energy-driven partitioning of signal processing algorithms in sensor networks. *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pp. 142–154, Springer Berlin Heidelberg, 2006.

[4] Y. Huang, et.al. Analysis of pkf: A communication cost reduction scheme for wireless sensor networks. *IEEE Transactions on Wireless Communications*, 15(2), pp. 843–856, 2016.

[5] C.-C. Shen, et.al. Energy-driven distribution of signal processing applications across wireless sensor networks. *ACM Transations on Sensor Networks*, 6(3), pp. 24:1–24:32, 2010.

[6] W. Yu, et.al. An altruistic compression-scheduling scheme for cluster-based wireless sensor networks. In *Proceedings of the 2015 IEEE Inernational conference on sensing, communication, and networking*, pp. 73–81, 2015.