DANCE: Distributed Application-aware Node Configuration Engine in Shared Reconfigurable Sensor Networks

Chih-Ming Hsieh, Zhonglei Wang and Jörg Henkel
Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
{chih-ming.hsieh, zhonglei.wang, henkel}@kit.edu

***Abstract*—Wireless sensor networks (WSNs) are often primarily tailored to single applications to achieve one specific mission. Considering that the same physical phenomenon can be used by multiple applications, the benefit of sharing the WSN infrastructure is obvious in terms of development and deployment cost. However, allocating the tasks to the WSNs to meet the requirements of all applications while keeping the energy efficiency is very challenging. Introducing reconfigurable nodes in the shared sensor networks can improve the performance, the energy efficiency and the flexibility but it increases the system complexity. In this paper, we propose a biologically inspired node configuration scheme in shared reconfigurable sensor network named DANCE, which can adapt to the changing environment and efficiently utilize WSN resources. Our experiments show that our scheme reduces the energy consumption by up to 76%.**

I. Introduction

Wireless sensor networks (WSNs) are traditionally designed for single applications with very specific requirements. Recently, the trend of WSN development has been moving from an application tailored system to an integrated infrastructure which can serve multiple applications, such like urban or environmental sensing systems [1], [2]. For instance, a greenhouse control system may utilize a WSN infrastructure to provide temperature and humidity monitoring, light control, security alarms and watering control. Obviously, a shared WSN can reduce the hardware and deployment cost by serving multiple applications with the same set of sensor nodes and the network comparing to deploying systems devoted to each application.

While WSNs are supporting an increasing number of applications, the tasks to be executed are also getting more complex. For example, advanced processing, such as image/audio recognition, becomes very common in wireless multimedia sensor networks (WMSNs) [3]. Reconfigurable platforms are able to perform complex tasks efficiently using hardware accelerators, while adapting to the changing requirements with the reconfiguration. However, the reconfiguration itself is an energy consuming operation [4] which should be managed properly in order to gain the benefits from the hardware acceleration.

With flash-based reconfigurable devices (e.g. Microsemi's IGLOO), it is possible to achieve low power operations at microwatt level in sleep mode [5]. Naturally, using such kind of reconfigurable platforms in shared sensor networks can provide even higher flexibility, performance and efficiency since the node can change their configuration according to the current requests corresponding to the applications being served. However, the issue of configuration management is even more challenging when it comes to a shared WSN infrastructure due to the various needs of the acceleration from different applications and the limited reconfigurable fabric of the sensor nodes. Taking smart building as an example, an FFT accelerator required by an acoustic event recognition and an AES accelerator required by a security alarm occupies 3031 (65.78%) and 2082 (45.18%) reconfigurable tiles, respectively, on a Microsemi A2F200M3F reconfigurable device [5], which is obviously not able to accommodate both accelerators and has to decide either not to serve one of the applications or to serve it without the acceleration. Since most application requests have random arrival rate and unpredictable duration, it is difficult, if not impossible, to come out with an optimal reconfiguration plan a priori.

978-3-9815370-0-0/DATE13/

To address aforementioned issues faced by the shared WSNs with reconfigurable sensor nodes, we propose a biologically inspired configuration management scheme named "Distributed Application-aware Node Configuration Engine" (DANCE). DANCE learns from the natural honey bees' forage behaviors and efficiently configures the sensor nodes based on a distributed decision making mechanism. Specifically, the key contributions of this paper include:

- We developed a novel configuration management scheme called DANCE for shared reconfigurable sensor networks, efficiently taking the application requirements into account. One advantageous feature of DANCE is to exploit the spatial correlation of sensor nodes to make decision of sensor node reconfiguration. This idea is inspired by honey bees' waggle dance for exploring and sharing profitable forage paths.
- DANCE enables self-organized application assignment and can adapt to the changing requirements. In DANCE the applications are deployed with respect to the efficiency of the hardware configuration.

The rest of the paper is structured as follows. In Section II an overview of the related work is presented. The system overview and the problem statement are given in Section III. In section IV, the details of system architecture are described before showing the experiment results in Section V. Finally, the paper is concluded in Section VI.

II. Related Work

Hinkelmann et al. [6] proposed a platform consisting of a RISC processor and a reconfigurable function unit to improve the energy efficiency of WSN applications. In [7], the authors exploit partially reconfigurable field programmable gate array (FPGA) to develop situation-based reconfiguration in WSNs. Depending on the requirement of the sensing accuracy, different processing modules can be configured to save the

energy. These investigations focus on the design of a single node without taking the performance of the whole network into account. In our previous work [4] we have investigated the benefits of run-time reconfiguration in clustered WSNs. A network level study was conducted to show how reconfiguration of sensor nodes improves the overall energy efficiency of clustered WSNs. Nevertheless, none of them studies the issues of reconfigurable nodes in shared sensor networks as we do in this work.

Application allocation and optimization in shared sensor networks has attracted the attention recently. The Task-Cruncher in [8] optimizes the sensing tasks by reducing redundant communication and computation efforts among multiple concurrent applications. [9] proposed a multi-application allocation and deployment system called UMADE. They introduce the concept of QoM (quality of monitoring) to characterize the performance of a sensing application and provide sensing data from suitable nodes to applications based on the QoM. The goal is to maximize the utility under the memory constraint of the sensor node. Task-Cruncher and UMADE target on application allocation for different problems such as, combining common tasks to reduce overhead or increasing system utility. As opposed to them, our work focuses on the configuration management problem where the efficiency will be affected by the application allocated. Therefore, in our reconfiguration scheme we take the application allocation into account to improve the energy efficiency of the whole network.

Biologically inspired algorithms imitate the natural systems and have been proposed to deal with various complex problems in computer science for many years [10]. For example, a honey bee inspired algorithm was proposed in [11] for dynamic server allocation in internet hosting centers. The authors model the internet server allocation problem with the honey bee forage mechanism to collect the maximal revenue from the service subscribers. Our work adopts the same forage mechanism for configuration management of applications in shared WSNs.

III. System Overview and Problem Statement

A. System Overview

A shared reconfigurable sensor networks consists of a resource-rich base station and resource-constrained reconfigurable sensor nodes. The reconfigurable sensor nodes are equipped with reconfigurable fabric which can load accelerators for certain operations required by the applications. Each node loads the accelerators according to the residing applications subject to the size of reconfigurable fabric. If certain accelerators can not be loaded, the node should decide whether to reconfigure the fabric or to fall back to the software implementation.

With the assistance of accelerators, the node can perform the tasks more efficiently. If a new task set needs to be performed, the node may reconfigure the fabric accordingly to achieve the maximal performance and amortize the energy overhead due to reconfiguration over the following cycles.

One characteristic of the sensor networks is that the operations or the sensor readings are often spatial correlated. The nodes sharing spatial correlation tend to perform similar operations to the same application and therefore have similar power consumption. This characteristic can be exploited to estimate the efficiency of the sensor node for the same type of the operations since the nodes currently executing the operations have the historical trace of the power consumption, which can be used as a reference to determine the efficient configuration.

B. System Objective

The objective of our system is to distribute the application's tasks to a set of sensor nodes so that the QoS requirement can be fulfilled and the energy efficiency can be maximized under the constraints of the sensor node. We formulate the problem as follows:

- Let $N = \{N_1, N_2, \cdots, N_m\}$ denote the m nodes composing the sensor network. Each node has a reconfigurable fabric which can accommodate a limited number of hardware accelerators. Let $H = \{H_1, H_2, \cdots, H_p\}$ be the total of p accelerators, and $h_k \subset H$ be the subset loaded to the node N_k. F_k and M_k denote the fabric and memory available at node $N_k, 1 \leq k \leq m$, respectively. A certain amount of energy is consumed by reconfiguration at node N_k and is represented by E_k. The fabric size required by h_k is denoted by $f(h_k)$.
- Let $A = \{A_1, A_2, \cdots, A_n\}$ denote the applications sharing the sensor network. Each application $A_i, 1 \leq i \leq n$ has memory requirement m_i, preferable accelerator list r_i and QoS requirement Q_i.
- Each application $A_i (1 \leq i \leq n)$ will be served by a set of nodes $S_i \subseteq N$ to achieve a QoS of $q_i(A_i)$. The energy consumption of the application A_i to achieve $q_i(A_i)$ is denoted by $e_i(q_i(A_i)) = \sum_{k=1}^{m} a_{ik} \times e_k(h_k)$, where a_{ik} is 1 if node N_k is serving application A_i and $e_k(h_k)$ is the energy consumption of node N_k configured with accelerator set h_k.

The total energy consumption of application set A is defined as the sum of the energy consumption of n applications. Therefore, the objective of the energy efficient allocation is to assign a set of nodes $S_i \subseteq N$ to serve application A_i so that

$$minimize\,(\sum_{i=1}^{n}\sum_{k=1}^{m}(a_{ik} \times e_k(h_k) + b_{ik} \times E_k)) \quad (1)$$

under the constraint of $\sum_{i=1}^{n} a_{ik} \times m_i \leq M_k$ and $f(h_k) \leq F_k$, where b_{ik} is 1 if node N_k performs the reconfiguration.

IV. Our DANCE System for Shared Reconfigurable Sensor Networks

Our DANCE system is inspired by the forage mechanism of honey bees to solve the problem of application allocation and configuration management in a self-organizing and energy-efficient way. Table I summarizes the similarities between DANCE and the honey bees' forage mechanism. For space reason, the table contains only a very brief description of honey bees' behaviors. For more details, please refer to [12]. In this section, we will describe our honey bee inspired algorithm in detail, following a description of primary components of the DANCE system, which is necessary for understanding the algorithm.

Table I. SIMILARITY BETWEEN DANCE SYSTEM AND HONEY BEE FORAGE MECHANISM.

DANCE in shared sensor networks	Honey bee forage mechanism
A set of m sensor nodes	A set of m nectar foragers
n applications generate tasks by QoS requirement	n flower patches with nectar supply
A group of sensor nodes takes tasks from the same application set	A group of foragers collects nectar at the same flower patch
Cost to serve an application's task depends the difference between the requested and the available accelerator set	Cost of forage trip will be dependent on the location of the flower patch
Reconfiguration for another accelerator set incurs time and energy consumption	Foragers spend time and nectar to learn new profitable flower patches
A neighboring node shares its configuration if it is energy efficient	A forager shares its foraging site via waggle dance if it is profitable
Based on the QoS requirement, an application provides the serving node a value-per-task-served	A flower patch provides nectar of various qualities

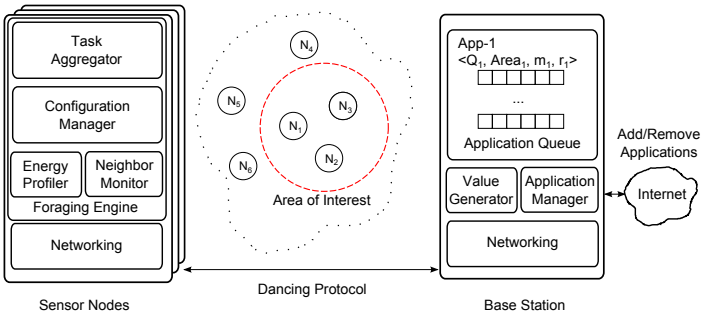

Figure 1. System Architecture: Users submit applications from internet. The sensor nodes exchange control information for application assignment and configuration management via dancing protocol.

A. System Architecture

Fig. 1 shows our DANCE architecture in shared reconfigurable sensor networks. **Application Manager** located on the base station manages the application requests and uses **Value Generator** to generate *value-per-task-served* indicating the reward (QoS gain) of serving an application. The sensor nodes can decide whether to serve this application based on *value-per-task-served*. **Energy Profiler** keeps an *efficiency value* (ε) representing the average energy consumption of current application set and hardware configuration. **Neighbor Monitor** maintains a list of neighbor nodes along with their *application list* (l), *efficiency value* (ε), and *accelerator list* (h). This information is broadcasted by the neighboring nodes which have more efficient configuration and will be used by the receiving sensor node to exploit the local spacial correlation. **Configuration Manager** collects the necessary information from other components and decide whether to perform a reconfiguration. For a reconfigurable node with p possible accelerators, a p-bit *accelerator list* $h =< b_1, b_2, ..., b_p >$ is maintained to indicate the available accelerators ($b_i = 1$ means accelerator i is loaded). **Configuration Manager** fetches the current efficiency value from **Energy Profiler** and compares it with the neighboring nodes who has the same *application list*. When **Configuration Manager** finds that its *efficiency value* is relatively low, it has a higher probability to perform a reconfiguration. **Task Aggregater** analyzes the current *application list* and schedules the shared sensing tasks for multiple applications. For all applications on the list, a maximal sampling rate should be choose to prevent missing samples.

B. Detailed Operations of the DANCE Scheme

Our DANCE system consists of four characteristics learned from honey bees to achieve self-organization and energy efficiency. First, the applications use *value-per-task-served* to attract the nodes to serve them (**postive feedback**). If too many nodes serve the same application, the value will drop and causes the node to find a new application to serve (**negaive feedback**). When a node learns the efficiency from the neighbors, it has a certain probability to follow its neighbors (**fluctuations**). Finally, the nodes share the efficiency with their neighbors to propagate better solutions (**multiple interactions**). The positive and negative feedback can deal with the changing application requests. Fluctuations provide the capability to discover potentially better efficiency and multiple interactions prevent complex centralized operations.

The operations of our DANCE scheme consist of two steps. First, the applications are assigned to the sensor nodes with the consideration of the QoS and the efficiency of the nodes' accelerator set. Second, the node learns the neighbors' efficiency of their accelerator set and decides whether to perform a reconfiguration.

1) Application assignment: In our DANCE scheme, the application assignment is initiated from the sensor node. Each sensor node sends a scout packet to the base station to collect the tasks from the application queue. Each task has *value-per-task-served* and the application queue accumulates *value-per-task-served* based on the QoS requirement. The selection of application i is based on a selection function $s(v_i, r_i, h) = \frac{v(Q_i)}{d(r_i,h)+1}$, where the function v denotes total *value-per-task-served* based on QoS Q_i, the function d calculates the Hamming distance of two p-bit vectors, r_i and h, indicating the requested accelerators and the available accelerators, respectively. The higher v indicates that the application requires more nodes to serve it (**postive feedback**) and the selection function assigns the application to a node with a similar accelerator set. For example, if application A_1, A_2, and A_3 have the same QoS requirement ($v = 60$) and require accelerator $r_1 =< 1,1,0,0,0,0,0,1 >, r_2 =< 0,0,1,1,0,0,0,0 >$, and $r_3 =< 0,0,0,0,1,1,1,1 >$, respectively, while a node N_k has $h =< 1,0,0,0,0,0,1,1 >$, the selection values will be $20, 12$, and 15, respectively. Therefore A_1 will be selected when the node searches for the application.

With the reply of the scout packet from the base station, a node determines the application list to be served. Task Aggregator schedules the sensing tasks and reports the sensor readings based on the aggregated sampling rate. The reporting packets get the replies from the base station regularly to update the profit value of the applications. If the profit value continuously drops below a certain threshold, this means that the application is currently served by enough nodes (**negative feedback**), the node may decide to drop current assignment and follow a neighboring node's assignment.

2) Configuration Management: After the first step, an application set whose required accelerator set is close to the available accelerator set on the local node will be assigned. To determine whether to keep the current accelerator set in re-

sponse to a scout packet, **Configuration Manager** calculates a probability value ($P_{conf} = \beta \cdot \frac{residual\ energy}{total\ energy}, \beta \in [0,1]$) based on the residual energy of the node. At this moment, the node may decide to reconfigure itself with a random accelerator set from required accelerators (**fluctuations**) if P_{conf} is high. This allows the node with high residual energy to explore configurations with higher efficiency. After sensing tasks start, **Energy Profiler** records the energy consumption of each cycle and calculates the *efficiency value* (ε). The node broadcasts a dance packet with its ε, *application list* (l), and *accelerators list* (h). In the meantime, the node checks the ε of the neighbors with the same l (**multiple interactions**). If its ε is below the average value of the neighbors, it may decide to perform reconfiguration.

V. Experiment and Evaluation

We use the Castalia network simulator [13] to simulate an area of 15m×15m with 4 different setups in terms of deployment density in order to study the effects of the spatial correlation. They are 1, 2, 4 and 8 sensor nodes in each 15m×15m area. The sink node is the same for all setups. Each node can load a limited number of accelerators from an accelerator set and has limited memory. Each accelerator in the accelerator set has a certain performance gain. The applications can arrive with a random arrival rate and duration or following a fixed schedule.

We compared four different approaches with our DANCE scheme. To provide a baseline of the comparison, we tested an approach without reconfiguration capability. Each node loads a maximum number of the accelerators from the whole accelerator set according to the ranking of the performance gain (highest first) independently of the application requirements. In contrast, Greedy-1 and Greedy-2 select the accelerators with highest performance gain from the union of the accelerator lists, r_i, requested by the applications. When a new set of applications is assigned, Greedy-1 and Greedy-2 load the new accelerators based on the performance ranking with a probability of 1.0 and 0.2, respectively. In addition, we also find out an optimal schedule by means of offline analysis. The simulation results with this optimal schedule are used to evaluate the energy efficiency of all the aforementioned schemes.

The greedy approaches choose the best set of the accelerator without the help of the neighbors. To show the problem of choosing only high ranking accelerators, we intentionally choose the applications, which frequently use the accelerators with low performance gain. In this scenario, the offline schedule knows the exact timing of reconfiguration and therefore yields the best results (6% of baseline energy consumption). Since Greedy-1 and Greedy-2 select the high ranking accelerators, the performance gain is not significant (45% and 66%, respectively) due to the low utilization of the selected accelerators. Because the utilization of the accelerators can not be known a priori, the greedy approaches have no means to prevent this situation. Our DANCE approach outperforms both greedy approaches and yields the results (8%) close to the optimized offline schedule because the node learns the configuration with better efficiency from its neighbors. Then we randomly generated the application sets with random arrival rates to simulate the real application behaviors. The normalized

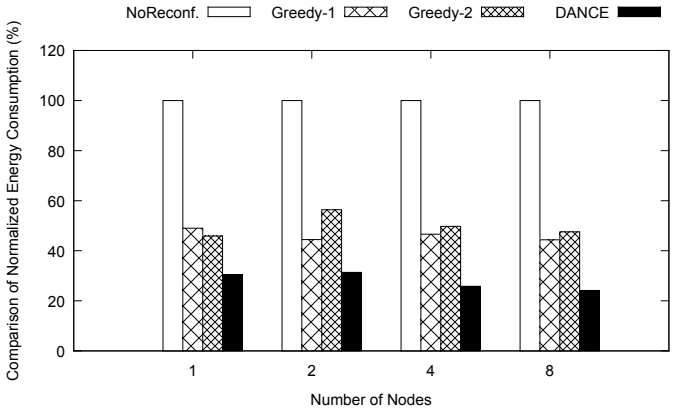

Figure 2. Normalized energy consumption with random application requests (optimal solution is not possible with random requests)

results are shown in Fig. 2. Our DANCE approach reduces 76% of the energy compared to the baseline and has 20% and 23% advantage compared to Greedy-1 and Greedy-2, respectively.

VI. Conclusions

In this paper, we investigated the configuration management problem on shared reconfigurable sensor networks. We first discussed the issues of the shared reconfigurable sensor network in terms of energy efficiency and then introduced our DANCE scheme inspired by the forage mechanism of honey bees. We evaluated our system by means of simulation at the network level and showed that our approach reduces the energy consumption by 76%, 20% and 23%, respectively, compared to non-reconfigurable and two greedy approaches.

References

[1] R. Murty, G. Mainland, I. Rose, A. Chowdhury, A. Gosain, J. Bers, and M. Welsh, "CitySense: an urban-scale wireless sensor network and testbed," in *HST'08*.

[2] M. A. Batalin, M. Rahimi, Y. Yu, D. Liu, A. Kansal, G. S. Sukhatme, W. J. Kaiser, M. Hansen, G. J. Pottie, M. Srivastava, and D. Estrin, "Call and response: experiments in sampling the environment," in *SenSys '04*.

[3] I. Akyildiz, T. Melodia, and K. Chowdury, "Wireless multimedia sensor networks: A survey," *Wireless Communications, IEEE*, vol. 14, no. 6, pp. 32–39, 2007.

[4] C.-M. Hsieh, Z. Wang, and J. Henkel, "A reconfigurable hardware accelerated platform for clustered wireless sensor networks," in *ICPADS'12*.

[5] "http://www.actel.com/products/."

[6] H. Hinkelmann, P. Zipf, and M. Glesner, "A domain-specific dynamically reconfigurable hardware platform for wireless sensor networks," in *ICFPT'07*.

[7] R. Garcia, A. Gordon-Ross, and A. George, "Exploiting partially reconfigurable FPGAs for situation-based reconfiguration in wireless sensor networks," in *FCCM '09*.

[8] A. Tavakoli, A. Kansal, and S. Nath, "On-line sensing task optimization for shared sensors," in *IPSN '10*.

[9] S. Bhattacharya, A. Saifullah, C. Lu, and G.-C. Roman, "Multi-application deployment in shared sensor networks based on quality of monitoring," in *(RTAS'10)*.

[10] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artif. Intell. Rev.*, vol. 31, no. 1-4, pp. 61–85, 2009.

[11] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in internet hosting centers," *Adaptive Behavior*, vol. 12, no. 3-4, pp. 223–240, Dec. 2004.

[12] T. D. Seeley, *The wisdom of the hive: the social physiology of honey bee colonies*. Harvard University Press, 1995.

[13] "http://castalia.npc.nicta.com.au/."