

# Evaluation of Cooperative Task Computing for Energy Aware Wireless Networks

Anders Brødløs Olsen, Frank H.P. Fitzek, Peter Koch  
Department of Communication Technology, Aalborg University  
Niels Jernes Vej 12, 9220 Aalborg Øst, Denmark  
Email: {abo|ff|pk}@kom.aau.dk

**Abstract**—In this paper we propose cooperation on task-set computation for cooperative groups of mobile wireless terminals. Energy consumption for future generation terminals is important, calling for new innovative concepts to alleviate the present evolution of evermore energy hungry terminals. Related to cooperative concepts a novel energy conservation method is proposed, which we will refer to as  $D^2VS$ . Our proposed method uses abstractions of traditional multi-processor environments, where terminals are considered as processing units connected by short range wireless networks. Energy conservation is obtained by the well known method of dynamic voltage scaling, which has proven to generate near optimum energy schedules of task-sets. Our simulation experiments show that up to 40% energy reduction on two cooperating terminals is obtainable. This is compared to a single energy aware terminal.

## I. INTRODUCTION

Wireless systems have become an essential part of our daily life. Recently, concepts on cooperation in wireless networks are introduced, with the overall purpose to join forces in order to reach a common QoS improvement. Existing wireless networks assume that each individual mobile terminal (likely owned by users who may not necessarily want to share their resources with other users) will follow pre-described protocols. However, users may modify behavior or protocols for self- or group-interests. Most already proposed cooperative methods are introduced to combat the inherent nature of wireless networks, where channel errors imply a significant issue, using basic principles to share resources for diversity. In [1] a number of physical layer cooperative issues are surveyed, and in broad terms relaying concepts are the main philosophy. Examples are; A) detect and forward, B) amplify and forward, and C) coded cooperation. In [2] cooperative networks at higher layers are discussed, outlining the cooperative network architecture. The key points are; systems should be layered on demand, reuse of module blocks, multiple services, and end-to-end connectivity across access technologies.

In Figure 1 the principle cooperative scenario for a cellular environment is illustrated. Terminals receive their application from the base station over the centralized wireless link, and then use a short range wireless protocol for cooperating on a joint enhancement of the application QoS. Cooperation is made by exchanging information locally using the short range link. An example, terminals suffers from fading and is therefore more likely to receive error prone data, however by cooperating on reception an overall improved reception is

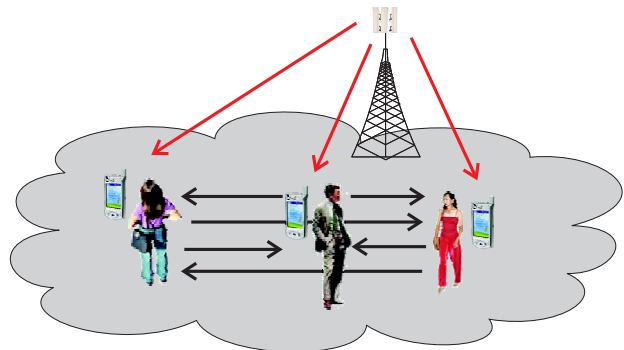


Fig. 1. Principle of cooperation between terminals within a cellular network.

likely to occur. It is evident that; capacity, spectral efficiency, bandwidth, reliability, and even security may be enhanced.

This work utilizes the cooperative concept for achieving *energy aware computing in cooperative wireless networks*. It is assumed that terminals applications can be abstracted into realtime task-sets, where tasks are defined by timing and workload specifications. Further more can task-sets be divided into sub-tasks, which can be distributed among terminals. The task distribution is performed over a short range wireless link, which forms a group of terminals cooperating on task execution. The individual terminals must be controlled by a power management system achieving energy reduction.

Our proposed method uses abstractions from traditional multi-processor architectures and scheduling, where terminals are considered as processing units and inter-connected with each other by short range wireless communication links. The cost of computing tasks on multiple processors, together with the communication overhead, must be considered. The principle approach is to; 1) employ various scheduling methods and optimizing cost functions for assigning tasks to multiple terminals, 2) distribute the tasks using the short range wireless network, and 3) reduce energy consumption on the individual terminals. As mentioned, energy control is an important issue and for single processors Dynamic Voltage Scaling (DVS) has proven to generate near optimal schedules for task-sets [3]. By utilizing task-set specification, arrival- and deadline-time together with task workload, DVS scales the processor performance by adjusting the clock frequency and supply voltage.

To the best of the authors knowledge, only one attempt on cooperative computing is made for wireless sensor networks [4]. In their work, an ILP-based approach is proposed together with a three-phase heuristic. The concept is derived for epoch-based real-time applications, with the results that system life times are improved by a factor of 10 in the best case. Conceptually, our work is similar, although it deviates in a number of essential ways: 1) In [4] tasks are described using Directed Acyclic Graphs (DAG), where we instead use the definition of real-time task-sets. Using DAG and multiple heuristic seems as a static approach of allocating tasks among nodes, where we eventually are targeting a more dynamic environment and scheduling mechanism. 2) From DVS literature, issues of task slack time reclaiming are shown to be important, and by static approaches this is not possible, where dynamic DVS methods are able to utilize runtime accumulation of task slack times for further energy reduction. 3) Finally, we believe that task allocation on other terminals not always is beneficial, which thereby call for dynamic decision mechanisms for task distribution.

In our work we propose a simulation environment where cooperative computing can be evaluated, using abstractions of multiple single processing units (terminals) connected in network architecture. In Section II our proposed method of cooperative task computing for energy aware wireless networks is discussed. Our approach of Distributed DVS, or D<sup>2</sup>VS for short, is made, outlining the principle concept of cooperative computing in wireless networks. In Section III the proposed simulation environment is presented, based on a previously proposed evaluation platform for single processor DVS evaluation. In Section IV the simulation results are presented, and finally Section V draws the concluding remarks.

## II. METHODOLOGIES

In Figure 2 a principle multi-processor architecture is illustrated. A multi-processor environment contains a number of processing units connected in a certain network topology. By this, mobile terminals are abstracted as processing units, which then is inter-connected by short range wireless links.

A number of overall assumptions are made: 1) The application is considered a task-set containing real time tasks. 2) Each

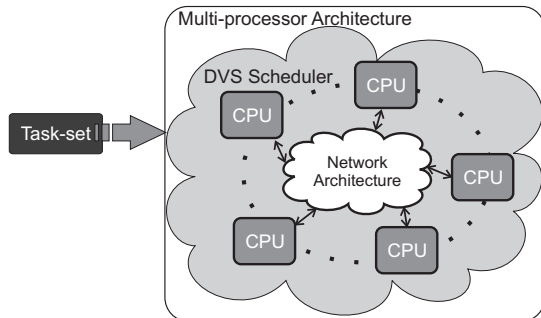


Fig. 2. Abstraction of a multiple processors environment, connected by a abstract network architecture. DVS schedules the multi-processors architecture and perform a energy conservation.

terminal can transmit and receive tasks (or parts hereof) from other terminals for computation. 3) A short range wireless infrastructure is available, able to handle transmission of tasks between terminals. 4) Efficient energy reduction methods are able to minimize the energy consumption. The energy control is both needed on the processing units (terminals), as well for the wireless network. Dynamic Voltage Scaling is used on the processors, in order to make the energy conservation. For the wireless network, an ideal energy aware protocol is assumed, able to effectively turn the network off whenever inactive.

The overall concept of operation is: *Mobile terminals cooperating on energy aware task execution must individually gain from the cooperation, and is therefore willing to exchange task-sets. If not fulfilled, terminals act selfishly and execute their task by them self.*

### A. Task-set specification

From traditional scheduling the notation of real-time task-sets is used. Only periodic task-sets are considered, where tasks are specified by:

- **Arrival time:** The time when a task becomes active.
- **Deadline time:** The time when a task must terminate its execution.
- **Task period:** The arrival frequency of the specific task. In scheduling theory, periodic arrival is typical used, due to simplicity. In practical systems, tasks also have a-periodic or burst like appearances.
- **Workload:** The workload introduced by the task on the processing unit, typically specified in Worst Case Execution Time (WCET). In scheduling, the actual functionality of the task is not important, but the time estimation for its execution time is.
- **Dependencies:** Determines the relations between tasks. In scheduling, independent task-sets are often used, but realistically tasks are likely to be connected in a given sequence. In this work only independent tasks are investigated.

There are mainly three real-time system definitions: 1) Systems with no real-time, where the quality of task execution is timeless. 2) Soft real-time, where task execution is limited by a deadline and a decaying cost function allowing for further task quality of delayed task execution. 3) Hard real-time systems, where tasks are limited by deadlines, having corrupted execution if not finished before its deadline.

For multi-processor scheduling, the parallelism in a task or task-set is important. This indicates how many processing units the task can be distributed onto. In this work two scenarios are considered: 1) Parallelism between tasks, implying that complete tasks are distributable. For independent task-sets full parallelism exists, where the number of tasks determines the maximum number of feasible terminals onto which the task-set can be distributed. 2) Tasks contain task parts that can be executed in parallel, implying that these sub-tasks are executable on different processing units.

## B. Task Distribution

The cost of task distribution is essential for the method, deciding upon the execution of a task on the local or at remote terminals. Task-sets are considered to belong to a given terminal, which refers to as the local terminal, where the terminals that can receive and execute tasks are referred to as remote terminals. The distribution decision is a runtime mechanism, which can be formulated as:

- $T$ : The Task-set
- $\tau_i$ : The  $i$ 'th task in  $T$ , containing  $1 \rightarrow n$  tasks
- $S_j$ : The  $j$ 'th sub-part of task  $\tau_i$ , having  $1 \rightarrow m$  parts
- $E(\cdot)$ : The convex energy operator for task execution

For the tasks in  $T$ , it is assumed that either tasks or sub-parts can be distributed among terminals. For the energy operator it is important to note that it has a convex shape, because of 1) the power consumption nature, and 2) since the terminals are using DVS methods to further reduce energy consumption. The overall distribution mechanism is divided into a number of functionalities. First, the energy cost of executing the entire task-set at the local terminal without cooperation:

$$E_{NoCoop} = E(T)$$

In the cooperative scenario there are two principle ways to distribute task. First, complete tasks are outsourced:

$$E_{Coop} = E_L \left( \tau_i \right)_{i:1 \rightarrow g} + E_R \left( \tau_i \right)_{i:g+1 \rightarrow n} + E_N \left( \tau_i \right)_{i:g+1 \rightarrow n}$$

where the energy operators  $E_L$ ,  $E_R$ ,  $E_N$  is the energy at 1) the local terminal, 2) the remote terminals, and 3) on the wireless network device, respectively. The energy consumption for executing some of the tasks locally and some remote, together with the overhead of transmitting the tasks to and from the remote terminal must be considered.

Secondly, the effect of distributing sub-tasks can be expressed as:

$$E_{Coop} = E_L \left( S_j, \forall \tau_i \right)_{j:1 \rightarrow h, i:1 \rightarrow n} + E_R \left( S_j, \forall \tau_i \right)_{j:h+1 \rightarrow m, i:1 \rightarrow n} + E_N \left( S_j, \forall \tau_i \right)_{j:h+1 \rightarrow m, i:1 \rightarrow n}$$

where all tasks are distributed, and  $S_j$  determines the sub-parts of the tasks that are executed on the local and remote terminals. For the distributed sub-parts, the network overhead must be evaluated, which eventually is a function of the sub-parts distributed to the remote terminal.

The overall philosophy of the two methods are essential identical, as workload is balanced among terminals. In the later all tasks are distributed as sub-tasks, where in the first only selected tasks are distributed among terminals. Distributing tasks or sub-tasks among the local and remote(s) terminals become a traditional optimization problem, with the challenge

of finding the minimum energy consumption by placing tasks on the available terminals.

The overall decision of executing all tasks locally or distributing them is a matter of choosing the cheapest energy consumption:

$$E = \min(E_{NoCoop}, E_{Coop})$$

The runtime mechanism making the described distribution decision is not within the scope of this work, but the overall prospect of the proposed methods will be evaluated using simulations.

## C. Energy Awareness

For battery powered systems, methods for energy reduction has been an intensive research topic within the last decade. Especially Dynamic Power Management (DPM) strategies are widely used to overcome the poor advance in battery technology. In [5] various DPM methods at different system levels are presented. On system level, the common method is to place system devices in power-down modes, whenever they are inactive. The recent advantages in embedded programmable processors are the support for dynamic changes to the clock frequency and supply voltage, a DPM technology referred to as Dynamic Voltage Scaling (DVS). The technology is adopted by the majority of the large manufactures on the processor marked, e.g., Intel with their SpeedStep® and AMD's PowerNow!™.

In wireless communication various energy aware protocols are proposed [6]. Energy conservation is typically considered at the physical layer, where a lot of work is made. However, work at higher protocol layers is getting more focus, with the principle of controlling behavior patterns for the network. In [6] methods for data link, network, transport, OS/middleware, and application protocol layers are surveyed and discussed.

DVS methods can be divided into two categories [7], Inter-task and Intra-task methods. The Intra-task methods use static analysis, typical graph analysis methods, providing voltage frequency scaling points for the task. Inter-task methods are a task-by-task method, typically an extension of traditional scheduling algorithms like; Fixed Priority (FP) and Earliest Deadline First (EDF). Of the two DVS categories Inter-task DVS are the most investigated, and also most practical.

In Figure 3 the basic principle of Inter-task DVS is illustrated. First of all, by the task specification the basic idea is to stretch task execution, so entire task periods are utilized. For Inter-task methods task-sets is scheduled, e.g. by using EDF as illustrated in bottom part of Figure 3. A typical task-set will cause a processor to idle, because of arrival pattern and workload specifications of the task-set. Inter-task DVS will then stretch task execution, based on a given algorithm as illustrated in Figure 3. The illustrated method uses an EDF schedule, stretching tasks when they appear uniquely active at the scheduling instant [8].

Most DVS policies proposed are for single processor environments [8], [9], [3], [10], where all, more or less, are using identical approaches in terms of; a known scheduling

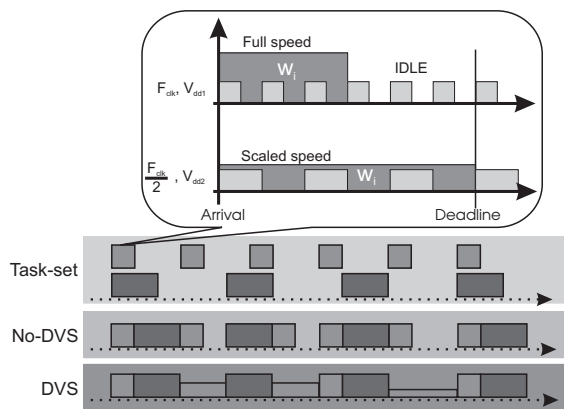


Fig. 3. Example task-set of two tasks with periodic arrival periods. When traditional schedules are used, slack time is introduced causing the processor to idle. DVS schedules stretch task execution time to utilize all available time, by scaling performance of the processor. (In the bubble: a single task defined by its parameters, showing the performance scaling.)

algorithm, together with the difference that the stretching and reclaiming of unused slack time is performed. For multi-processor environments significantly less work is proposed. One common approach is to evenly distribute task on the available processing units according to workload, a method also known as load balancing [11].

In order to understand the effort of stretching task execution time, an understanding of energy consumption is needed. The power dissipation for typical embedded implementation technology, which is CMOS, contains three components: 1) A *dynamic* power consumption relating to the switching activity of the circuitry. 2) A *leakage* power consumption which represents the constant flow of current in the reversed-biased diode of the individual transistors. 3) Finally, the *short* power which is the direct path established from the supply to ground whenever the circuit makes transitions. Of the three dynamic power is the most significant and expressed by,

$$P_{dynamic} = K \cdot C_L \cdot V_{dd}^2 \cdot f \quad (1)$$

where  $K$  is the activity factor of the circuitry,  $C_L$  the equivalent load capacitance,  $V_{dd}$  the supply voltage, and  $f$  is the clock frequency. Another important physical factor is the circuitry delay, expressed by,

$$T_{delay} = k \frac{V_{dd}}{(V_{dd} - V_{th})^\alpha} \propto \frac{1}{V_{dd}} \quad (2)$$

where  $k$  and  $\alpha$  are technology constants, and  $V_{th}$  is the threshold voltage. The time delay is typically considered proportional to the inverse of the supply voltage, meaning that; lower  $V_{dd}$ , larger delay, and thus lower obtainable clock frequency. This implies that optimal pairs of supply voltages and clock frequencies can be found.

These two physical parameters are the main reason why DVS is very effective. Results for evaluating this by practical measurements on a high performance scalable processor is shown in Table I. At the left hand side, measurements of processor speeds from 1/1 (highest clock frequency) to 1/4

speed is shown. For the specific example, the effect of reducing the speed by a factor of two is providing approximately 3.5 times power reduction. At the right hand side, the effect of combining the processor into a network of processors is illustrated using 1 to 4 processors. Assuming now that workload can be divided evenly on processing units, and also that relative identical performance of the systems is obtained. This implies that having multiple slow operating units may lead to an energy gain, compared to a single fast unit. From Table I it is also seen that the number of multi-processors has an impact on the minimum energy consumption. This suggests that a few slow units are providing considerable energy savings as compared to a single processor solution. For the specific example, 3 processing units have the minimum energy consumption, with a gain of 45% compared to a single processor.

TABLE I  
POWER MEASUREMENTS ON AN ANALOG DEVICES BLACKFIN  
PROCESSOR (BF533)

Processing speed	Power	Processing units	Power
@ 1/1	275 mW	# 1	275 mW
@ 1/2	80 mW	# 2	160 mW
@ 1/3	50 mW	# 3	150 mW
@ 1/4	45 mW	# 4	180 mW

### III. SIMULATION ENVIRONMENT

The simulation experiments in this paper are based on the tool proposed in [12], with the modification of task distribution and multiple single processors. Due to simplification only two terminals are considered, as illustrated in Figure 4.

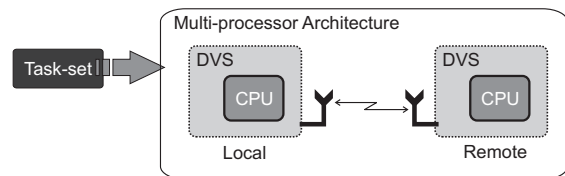


Fig. 4. Experimental setup having a local DVS scalable processor (terminal) and a single remote processor connected by a wireless link.

The task-set is defined as described in Section II-A, with periodic arrival times, deadline times identical to task periods, and constant task workloads noted in worst case execution times. For the experiments, randomly generated task-sets are used. The random task-sets are generated using bounds, e.g., arrival times are defined by a minimum and maximum time, and a specific arrival times for the tasks are chosen uniformly. In the experiments, the bounds are chosen identical to the reference task-sets from [8], and workload defined by a task-set utilization value, where the individual tasks are assigned a random workload, adding up to the utilization factor.

The network overhead is important for these experiments, using the following assumptions. From Section II-B two different task distribution strategies are explained; one where non-partitioned tasks are distributed and the other where only

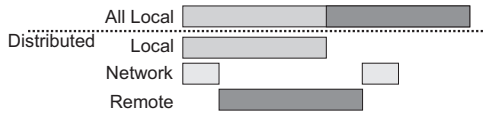


Fig. 5. Example of sub-task distribution, showing the overheads introduced on the network. Where identical pieces are executed on the local and remote terminals, introducing loads on the network when the sub-task is transmitted to and from the remote terminal.

sub-parts are distributed. The network overhead using sub-tasks is illustrated in Figure 5. The task, for this example, is divided into equal sized pieces, where one is executed at the local terminal and the other on the remote terminal. It is assumed that the network technology can transmit tasks, while the local terminal executes tasks. At the network, load is introduced determined by the data size of the distributed task, and introducing a load when the task is transmitted from the local terminal and when received from the remote terminal. The network load is defined by a percentage of the task execution time, introduced when transmitting as well as receiving the task. The network will introduce an energy overhead determined by the active time of the network, as well as a time delay eventually compromising the throughput of the application. For this work, the overheads contributed by the distribution mechanism are considered negligible. If this overhead should be accounted for, it will displace the local task by the size of the overhead, and naturally also the network task and arrival time at the remote terminal.

In Figure 4 DVS methods are utilized at both terminals, where these are considered as single processors connected in a wireless network link. The DVS scaling conducted is using a well known method proposed in [8] called lppsEDF. The method is not optimal, but due to its simplicity easy to modify for our purpose. The method uses pre-knowledge of the task-set, where a reference speed is determined for the task-set. This reference frequency is adjusted according to our pre-knowledge of the distribution ratio between the local and the remote terminal.

For the simulations, homogeneous terminals are assumed, meaning that terminal specification, e.g., energy dissipation, and network specification are similar. The energy consumption is modeled by a simple squared speed function, typically used in the DVS literature. Normalized speeds from 1 to 0, with intervals of 0.1 are used. For the network, the power is assumed to be constant, within the range of 1 to 0, meaning that 1 is identical energy consumption when the processor runs at maximum speed. Normalized values are chosen as the ratio between processor and network energy consumption is of interest.

#### IV. SIMULATION RESULTS

##### A. Task Distribution

The workload distributed among the local and remote terminals, together with the network overhead will have an impact on the performance of the proposed method. In these experiments, randomly generated task-sets with network loads

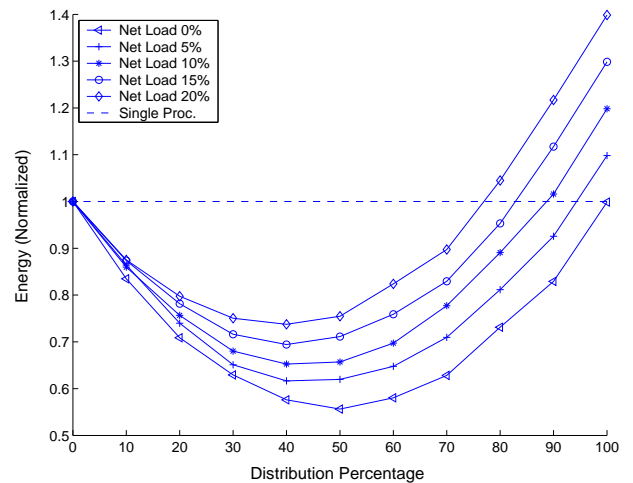


Fig. 6. Effect of task distribution, where at the left hand side is an all local execution, indicated by a zero percentage distribution. At the right hand side an all remote execution. Shown at different network loads.

of  $\{0\%, 5\%, 10\%, 15\%, 20\%\}$ , and ten tasks (fully utilizing the processor if executed at the local terminal) are employed. Non-partitioned tasks are distributed, as the ratio between local and remote execution are the target for the experiment. The network power is set to one, which is identical to the terminal power when the task-set is executed at a single terminal.

In Figure 6 the results are plotted showing the effect of network load caused by the tasks and the ratio between executing tasks local or remote. The results show that for zero network overhead, the minimal energy consumption is when the task-set is split fifty-fifty among the two terminals. This relates to traditional scheduling, where load balancing is shown to be optimal for systems neglecting communication overheads [11]. The higher the network load, the optimum point of distributed task seems to move left. As expected, this means that it will be more efficient to execute a larger portion of the tasks locally.

The optimal ratio between local and remote execution can mathematically be expressed as;

$$\min (U_L^2 + U_R^2 + (N_L(U_R) \cdot N_P))$$

where the combined task-set utilization is  $U = U_L + U_R$ , and  $U_L$  are the utilization at the local terminal, and  $U_R$  at the remote terminal. Terminal utilization is expressed using a traditional earliest deadline first utilization expressing  $U = \sum \frac{W_i}{P_i}$ , where  $W_i$  is workload and  $P_i$  is period of task  $i$ . The squared utilizations are inherited from the power relation, which can be model as speed squared.  $N_L$  is the network load caused by distributed tasks  $N_L = \sum n_{load}(\tau_i)$ , where  $n_{load}$  is the network load of the task, which is a function of the distributed tasks.  $N_P$  is the power consumption by the network.

##### B. Number of Tasks in the Set

Distributing tasks fifty-fifty among the local and remote terminal, the effect of the number of tasks in the task-set and

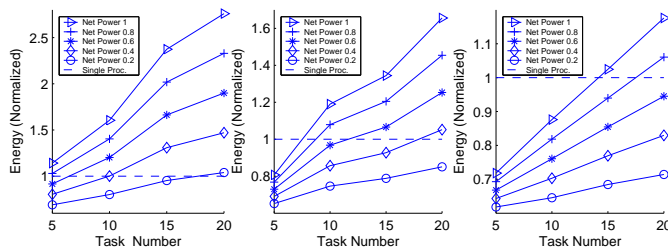


Fig. 7. Number of task in the task-set, shown at different task set utilizations {0.5, 0.75, 1}, and with network powers of {1, 0.8, 0.6 0.4 0.2}. It should be noted that different scales on the y-axis are used.

network power are next investigated. These experiments are made with {5, 10, 15, 20} tasks, network powers of {1, 0.8, 0.6 0.4 0.2}, and task-set utilization of {1, 0.75, 0.5}. For the experiments, network loads are defined by a fixed constant for every task, instead of a percentage as previously used. The constant is chosen to be identical to 2% of the shortest task period (50 time units). The reason for this fixed workload is that a percentage, which is a function of the task-set utilization, will result in similar energy measures, and therefore not show the impact of task number and network power.

In Figure 7 the results are shown. The first observation is that task-set utilization is having large influence on the performance of the proposed method. When a task-set utilization of 0.5 is used, it seems better to execute all tasks locally, for the majority of the cases. Only for a low network power gains are obtained, and only for task-sets with a few tasks. The same trend for task-set utilization 0.75 is seen, where both network power and number of tasks are having influence. For task-set utilization 1 it is clearly seen that both higher network power and also number of task are resulting in energy gains from our proposed D<sup>2</sup>VS method.

From the experiment it is clearly seen that task-set utilization and network power is affecting our proposed method. The number of tasks influences the result as an increasing overhead on the network is introduced. The network power is obviously increasing this effect, meaning that for low network power a large overhead can be supported, but as the network power increases the joint energy exceeds the consumption of an all local execution. The effect of task-set utilization is also related to energy dissipation in circuitry. At high utilization, the processor runs at a high speed, suffering from the convex nature of energy consumption, therefore also resulting in large energy gains. At lower utilization, this gain is becoming less, meaning that the overhead of distributing tasks becomes the dominant factor. This also indicates that a careful distribution of tasks must be made.

## V. CONCLUSIONS AND FUTURE WORK

In this work we have introduced the concept of "Distributed DVS", or D<sup>2</sup>VS for short, for cooperative task computing in wireless networks. We abstracted terminals into processing units and thereby proposed to use D<sup>2</sup>VS to decrease the overall energy consumption for each mobile terminal. The concept

is based on the fact that the terminal applications can be abstracted into task-sets, which are defined by arrival- and deadline-times, and task workloads. Also assuming inherent parallelism enabling division of tasks or sub-part hereof, allowing execution on different terminals. Terminals are connected by short range wireless communication links, allowing for tasks to be distributed among terminals.

For cooperation on task-set execution it is important that each individual terminal benefit from the cooperation, but also that an overall energy reduction is obtained. By mean of our simulation results we can show that D<sup>2</sup>VS introduces energy reduction of approximately 40% in the best case, but also that careful and wisely distribution of the task must be made, according to system specifications. We conclude that D<sup>2</sup>VS is a new effective method to overcome the increasing energy consumption, and thereby being able to break the energy trend for future wireless networks

The future steps of this work include the study of the numerous number of adjustable system parameters. The most obvious suggestion is to introduce a runtime mechanism for task distribution, using cost functions which models terminal energy consumption and the wireless networks overheads.

## REFERENCES

- [1] A. Nosratinia, T. Hunter, and A. Hedayat, "Cooperative communication in wireless networks," in *Communications Magazine, IEEE, Vol.42, Iss.10*, 2004, pp. 74–80.
- [2] C. Politis, T. Oda, S. Dixit, A. Schieder, K.-Y. Lach, M. Smirnov, S. Uskela, and R. Tafazolli, "Cooperative networks for the future wireless world," in *Communications Magazine, IEEE, Vol.42, Iss.9*, 2004, pp. 70–79.
- [3] H. Aydin, R. Melhem, D. Mosse, and P. Alvarez, "Dynamic and aggressive scheduling techniques for power-aware real-time systems," in *Proceedings of Real-Time Systems Symposium (RTSS'01)*, 2001, pp. 95–105.
- [4] Y. Yu and V. Prasanna, "Energy-balanced task allocation for collaborative processing in wireless sensor networks," *Mobile Networks and Applications*, vol. 10, no. 1-2, pp. 115–131, 2005.
- [5] L. Benini and G. de Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Academic Publishers, 1998.
- [6] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. Chen, "A survey of energy efficient network protocols for wireless networks," *Wireless Networks*, vol. 7, no. 4, pp. 343–358, 2001.
- [7] W. Kim, D. Shin, H. S. Yun, J. Kim, and S. L. Min, "Performance comparison of dynamic voltage scaling algorithms for hard real-time systems," in *Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*, 2002, pp. 219–228.
- [8] Y. Shin, K. Choi, and T. Sakurai, "Power optimization of real-time embedded systems on variable speed processors," in *Proceedings of International Conference on Computer-Aided Design (ICCAD'00)*, 2000, pp. 365–368.
- [9] P. Pillai and K. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Operating Systems Review (ACM), 18th ACM Symposium on Operating Systems Principles (SOSP01)*, 2002, pp. 89–102.
- [10] W. Kim, J. Kim, and S. Min, "A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis," in *Proceedings of the Design Automation and Test in Europe (DATE'02)*, 2002, pp. 788–794.
- [11] H. Aydin and Q. Yang, "Energy-aware partitioning for multiprocessor real-time systems," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, 2003, pp. 113–121.
- [12] A. B. Olsen, F. Buttner, and P. Koch, "On combined dvs and processor evaluation," in *Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation (PATMOS'04)*, 2004, pp. 322–331.