Energy Efficient Task Allocation over Mobile Networks

Carmela Comito
DEIS, University of Calabria
Rende (CS), Italy
ccomito@deis.unical.it

Deborah Falcone
DEIS, University of Calabria
Rende (CS), Italy
falcone@si.deis.unical.it

Domenico Talia
ICAR-CNR
DEIS, University of Calabria
Rende (CS), Italy
talia@deis.unical.it

Paolo Trunfio
DEIS, University of Calabria
Rende (CS), Italy
trunfio@deis.unical.it

***Abstract*—In this paper we present an Energy-Aware Scheduling strategy that assigns computational tasks over a network of mobile devices optimizing the energy usage. The main design principle of our scheduler is to find a task allocation that prolongs network lifetime by balancing the energy load among the devices. We have evaluated the scheduler using a prototype of the system that includes smart phones and Android emulators. Experimental results show that significant energy savings can be achieved by using our energy-aware scheduler compared to classical time-based scheduler, while meeting the specified performance constraints.**

***Index Terms*—Mobile computing; Energy efficiency; Task Scheduling.**

I. Introduction

A mobile ad hoc network (MANET) is a self-configuring network of mobile devices connected by ad hoc wireless links and equipped with networking capabilities. Recent developments in the technologies of laptops and PDAs together with the reduction of their costs have incredibly raised the interest in MANETs.

For mobile devices running on batteries, energy efficiency is a key concern to enable effective and reliable computing over them. Efficient resource allocation and energy management can be achieved through clustering of mobile nodes into local groups. Clustering the network promotes and eases collaborations among mobile users. We refer to a cooperative MANET architecture where mobile devices are organized into local groups (also termed clusters or mobile groups).

Such a cooperative MANET can be seen as a system including a set of consumers, e.g. mobile applications or tasks, and a set of resources as energy and processing power. To make the most of all available resources, a proper distribution of tasks among the devices such as to optimize the energy consumption, represents a key issue to be addressed. The design and implementation of such task allocation (or task scheduling) strategy is the objective of this paper. The main design principle of our scheduler is to find a task allocation in order to prolong network lifetime. We, thus, propose an energy-aware task allocation scheme that distributes energy consumption among clusters by balancing the energy load among them. To this aim, we designed and implemented a two-phase heuristic-based algorithm. This algorithm first tries to assign a task locally to the cluster that generated the execution request by maximizing the cluster residual life. If the task cannot be assigned locally, the second phase of the algorithm is performed by assigning the task to the most suitable node all over the network of clusters, maximizing this way the overall network lifetime. We characterize the energy consumption of mobile devices defining an energy efficiency model in which the energy costs of both computation and communication activities are taken into account.

There are several works in literature whose goal is minimizing the overall energy dissipation of the system. However, this goal does not capture the nature of cooperative ad-hoc systems. The reason is that minimizing the overall energy dissipation can lead to heavy use of energy-effective devices, regardless of their remaining energy. The consequent short lifetime of such devices will very likely compromise the system performance. This weakness is a major motivation of the proposed energy-balanced task allocation scheme.

We have evaluated our scheduler using a prototype of the system that includes smart phones and Android emulators. The experimental results show that a significant improvement can be achieved using our energy-aware scheduler compared to the time-based round robin scheduler. In details, our algorithm: (i) is effective in prolonging network lifetime by reducing the energy consumption; (ii) is able to complete a greater number of tasks in the same experimental settings; (iii) in all the experiments performed, is able to keep alive all the devices thanks to its energy load balancing scheme. Our approach, thus, enhances the energy-efficiency of the system compared to classical time-based scheduling algorithms like round robin.

The remainder of the paper is organized as follows. Section II presents the reference mobile ad-hoc network architecture. Section III describes the energy model. Section IV presents the energy-aware task allocation scheme and related algorithms. Section V presents the experimental results. Section VI discusses related work. Finally, Section VII concludes the paper.

II. System Architecture

We consider a multi-hop wireless ad-hoc mobile network architecture. In a wireless mobile ad-hoc network, which changes its topology dynamically, efficient resource allocation, energy management and routing can be achieved through

adaptive clustering of the mobile nodes. In a clustering scheme the mobile nodes are divided into virtual groups. Generally, geographically adjacent devices are assigned to the same cluster. Under a cluster structure, mobile nodes may be assigned a different function, such as *cluster-head* or *cluster member*. A cluster-head normally serves as the local coordinator for its cluster, performing intra-cluster transmission arrangement, data forwarding, and so on. A cluster member is usually called an ordinary node, which is a non cluster-head node without any inter-cluster links.

We refer to the cooperative system, referred to as Mobile-to-Mobile (M2M) architecture, depicted in Figure 1, designed to allow on-demand collaborations among mobile nodes. Examples of mobile-to-mobile collaborations occur in several domains such as disaster relief, construction management and healthcare. In order to promote and ease collaborations when two or more mobile users, who are members of the same organization or simply collaborate, meet each other, we let them grouping into clusters referred to as *mobile groups*. Consequently, the proposed architecture includes a number of mobile groups or clusters. Figure 1 shows the interactions

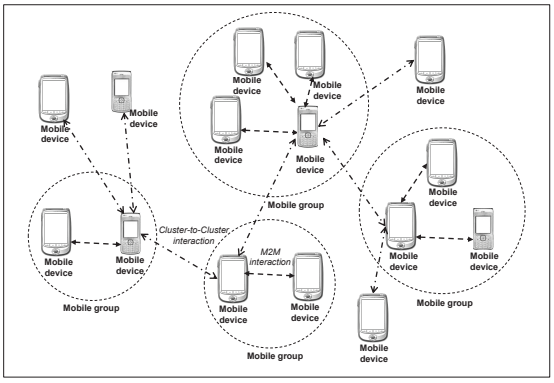

Fig. 1. The M2M system architecture. The arrows denote remote service calls.

among the different components of the architecture. Mobile nodes within a group interact trough ad-hoc connections (e.g., wi-fi, bluetooth) that we refer to as *M2M connections*, represented as dotted arrows in Figure 1. Interactions among mobile groups (*cluster-to-cluster connections*) take place through ad-hoc connections among the cluster-heads of the groups and are represented as dot-dash arrows. All types of interactions take place either to ask for a computation request or to cooperate in order to collaboratively execute a computational task.

This paper focuses on an energy-aware scheduling strategy allowing to efficiently allocate tasks over the clustered M2M architecture. More details about the clustering scheme can be found in our previous work [1].

III. Energy model

Energy consumption of mobile devices depends on the computation and the communication loads. We define $\mathrm{E_i}$ as the rate of energy consumption of node i in the time interval δt, which is the sum of all energy consumption for communication, $\mathrm{ET_i}$, and computation, $\mathrm{EC_i}$, of all the tasks assigned to node i within the time interval δt:

$$\mathrm{E_i} = \mathrm{EC_i} + \mathrm{ET_i} \tag{1}$$

Our approach is to estimate the energy consumption for computation and to analytically evaluate the energy consumed for communication. This last issue is the main aim of the section.

In MANET networks, nodes must always be ready to receive traffic from neighbors due to the absence of base station nodes. Thus, a network interface operating in ad-hoc mode has to continuously listen to the wireless channel, consuming this way a constant idle energy power. Therefore, each node *overhears* every packet transmission occurring in its transmission range consuming this way energy uselessly. This idle energy consumption is referred to as *overhearing*. Due to overhearing, a new cost in the computation of per-packet energy consumption is introduced and it is the cost for discarding overheard packets. Therefore, to model the energy consumed for communication, the costs to send, receive and discard a packet must be included. Consequently, the energy consumed by a node i for communication can be defined by the following equation:

$$\mathrm{ET_i} = \mathrm{E_{send_i}} + \mathrm{E_{receive_i}} + \mathrm{E_{discard_i}} \tag{2}$$

A packet may be sent through a broadcast or a point-to-point channel. With the former mode the packet is received by all hosts within the sender's transmission range; whereas with the latter mode it is discarded by non-destination hosts.

According to [3] we evaluate the energy consumption behavior of the mobile ad hoc network based on a model where the cost for a node to send or receive a message is modeled as a linear function. In this function there is a fixed cost associated with channel acquisition and an incremental cost proportional to the size of the message. The fixed channel access costs, denoted as $\mathrm{b_{send}}$ and $\mathrm{b_{recv}}$, and the incremental costs, $\mathrm{m_{send}}$ and $\mathrm{m_{recv}}$, are the same for broadcast and point-to-point. For point-to-point traffic, the fixed cost includes also the MAC negotiation. In the IEEE 802.11 MAC protocol, the source sends an RTS (request-to-send) control message, identifying the destination. The destination responds with a CTS (clear-to-send) message. Upon receiving the CTS, the source sends the data and awaits an ack from the destination. For simplicity, these small control messages are assumed to have the same fixed send ($\mathrm{b_{sendctl}}$) and receive ($\mathrm{b_{recvctl}}$) costs.

The cost, $\mathrm{E_{send_{ij}}}$, for a node i to send a point-to-point packet to a node j is described by the following equation:

$$\mathrm{E_{send_{ij}}} = \mathrm{b_{sendctl}} + \mathrm{b_{recvctl}} + \mathrm{m_{send}} * |\mathrm{MSG}| + \mathrm{b_{send}} + \mathrm{b_{recvctl}} \tag{3}$$

where $|MSG|$ is the size (number of bits) of the message exchanged among nodes i and j.

The cost to receive a point-to-point packet is modeled through the following equation:

$$\mathrm{E_{rec_{ij}}} = \mathrm{b_{recvctl}} + \mathrm{b_{sendctl}} + \mathrm{m_{recv}} * |\mathrm{MSG}| + \mathrm{b_{recv}} + \mathrm{b_{sendctl}} \tag{4}$$

In case of broadcast transmission, the cost to send a packet is represented through equation 5 while the cost to receive a

packet is given by equation 6:

$$\mathrm{E}_{\mathrm{send}_{\mathrm{broad}_\mathrm{i}}} = \mathrm{m}_{\mathrm{send}} * |\mathrm{MSG}| + \mathrm{b}_{\mathrm{send}} \quad (5)$$

$$\mathrm{E}_{\mathrm{rec}_{\mathrm{broad}_\mathrm{i}}} = \mathrm{m}_{\mathrm{recv}} * |\mathrm{MSG}| + \mathrm{b}_{\mathrm{recv}} \quad (6)$$

Thus, the energy cost of node i for sending (equation 7) and receiving (equation 8) packets depends on the used transmission mode:

$$\mathrm{E}_{\mathrm{send}_\mathrm{i}} = \begin{cases} \mathrm{E}_{\mathrm{send}_{\mathrm{ij}}} \text{ if point-to-point} \\ \mathrm{E}_{\mathrm{send}_{\mathrm{broad}_\mathrm{i}}} \text{ otherwise} \end{cases} \quad (7)$$

$$\mathrm{E}_{\mathrm{receive}_\mathrm{i}} = \begin{cases} \mathrm{E}_{\mathrm{rec}_{\mathrm{ij}}} \text{ if point-to-point} \\ \mathrm{E}_{\mathrm{rec}_{\mathrm{broad}_\mathrm{i}}} \text{ otherwise} \end{cases} \quad (8)$$

Non-destination nodes within the transmission range of either the transmitting or receiving nodes overhear the traffic. The cost of discarding is comparable to the one of a broadcast receiving:

$$\mathrm{E}_{\mathrm{discard}_\mathrm{i}} = \mathrm{m}_{\mathrm{discard}} * |\mathrm{MSG}| + \mathrm{b}_{\mathrm{discard}} \quad (9)$$

IV. The Energy-aware scheduler

In this section we present an energy-efficient dynamic task allocation strategy over the cooperative M2M architecture. In such an approach whenever a task has to be executed, an efficient task assignment is found such that the total consumed energy in the network is minimized and, thus, the network lifetime is prolonged.

A. Scheduling Model

The scheduling problem is the process of mapping a given application onto a target architecture by: (1) selecting which task of the application shall be considered; (2) allocating this task to a resource; (3) computing start and execution times for the task; (4) repeating these steps until all tasks are scheduled. It is common in the literature to use the terms task allocation and task scheduling interchangeably. However, scheduling is commonly used to describe all of the above mentioned steps as well as to describe the computation of start and execution times only. Task allocation is, therefore, a step of the more general scheduling problem; it can also be seen as a global scheduling or meta-scheduling that distributes the tasks among the devices. Once tasks have been allocated, the problem becomes one of defining a feasible local schedule that manages task execution for each node. In this paper we focus on the task allocation problem and we refer to task allocation or task scheduling interchangeably.

We introduce the following model to support the description of the scheduling strategy.

$\mathcal{D}= \{\mathrm{d}_1, \mathrm{d}_2, ..., \mathrm{d}_\mathrm{m}\}$ is the set of devices in the system. A device is described in terms of:

- battery level;
- processing capability;
- time, memory and energy consumed for computation;
- time and energy consumed for communication.

We refer to a task model with independent tasks, either atomic applications or tasks without dependencies. We denote with $\mathcal{T}= \{\mathrm{t}_1, \mathrm{t}_2, ..., \mathrm{t}_\mathrm{n}\}$ the set of tasks to be executed. A generic task $\mathrm{t_i}$ is characterized by the following features:

- the amount of data to be processed;
- execution time of task $\mathrm{t_i}$ on a device $\mathrm{d_j}$ over a data set of size s;
- energy consumption of task $\mathrm{t_i}$ on a device $\mathrm{d_j}$ over a data set of size s;
- memory consumption of task $\mathrm{t_i}$ on a device $\mathrm{d_j}$ over a data set of size s;

Before going into the details of the scheduling policy, some definitions and notations are introduced to support the proposed scheduler.

- $\mathrm{PC_i(t)}$: processing capacity of device $\mathrm{d_i}$ at time t.
- $\mathrm{M_i(t)}$: memory availability of device $\mathrm{d_i}$ at time t.
- $\mathrm{EEC_i(t_j, s)}$: estimated energy consumed for computation by device $\mathrm{d_i}$ to process a task $\mathrm{t_j}$ over a data set of size s.
- $\mathrm{EET_i(t_j, s)}$: estimated energy consumed for communication by device $\mathrm{d_i}$ to process a task $\mathrm{t_j}$ over a data set of size s.
- $\mathrm{EMC_i(t_j, s)}$: estimated memory consumption of device $\mathrm{d_i}$ to run a task $\mathrm{t_j}$ over a data set of size s.
- $\mathrm{EPC_i(t_j, s)}$: estimated processing capacity required by device $\mathrm{d_i}$ to execute a task $\mathrm{t_j}$ over a data set of size s.

Definition 1: Let be $\mathrm{RE_i(t)}$ *the residual energy available at node* i *at time* t*, and* $\mathrm{P_i(t)}$ *the instantaneous power; the residual life of node* i *at time* t*,* $\mathrm{RL_i(t)}$*, is defined as follows*:

$$\mathrm{RL_i(t)} = \mathrm{RE_i(t)}/\mathrm{P_i(t)}. \quad (10)$$

The classical task allocation problem can be reformulated here as the problem of finding the proper task assignment that minimizes the energy dissipated in the system and can be defined as follows:

Definition 2: Given a task model T *and a device model* D*, determine a task allocation* TA *that maps each task to a device such as to maximize the network lifetime,*

where the network lifetime can be defined as the period from the beginning of the application execution to the time when no enough devices are alive to deliver the required performance.

B. Scheduling Strategy

This section presents an energy-aware (EA) dynamic task allocation scheme over a cooperative MANET able to deal with a system where independent tasks arrive at each node dynamically over time.

The task allocation problem has been proven to be NP-Complete in its general form [2]. However, some optimal algorithms have been proposed for some restricted versions of the problem and some heuristic-based algorithms have been proposed for the more general versions of the problem allowing to find good allocations in polynomial time [4]. Thus, the problem is finding the minimum cost task allocation.

We propose a two-phase heuristic-based, decentralized algorithm. When an assignment decision has to be made for a task, the first phase, referred to as *local assignment* phase, is responsible for local task arbitration: it considers the energy

consumption of task execution on the different devices within the local cluster. The algorithm tries to minimize the total consumed energy in the cluster by assigning the task to the device that allows to extend the cluster residual life. If the first phase is not feasible, the second phase, referred to as *global assignment* phase, is responsible for task arbitration among clusters: the task will be assigned to the most suitable device, all over the network of clusters, that maximizes the overall network lifetime.

We formalize the problem of task allocation as an optimization problem. As said before, the aim of the optimization is to maximally extend the life of all the nodes in the network by balancing the load proportionally to the energy of each node such as to maximize the network lifetime. We optimize the problem by iteratively trying to improve a candidate solution. A feasible allocation is optimal if the corresponding group residual life (in case of local assignment) or system lifetime (in case of global assignment) is maximized among all the feasible allocations.

The candidate nodes to which a task $\mathrm{t_a}$ could be assigned have to satisfy the following constraints:

1) a node $\mathrm{d_i}$ must have enough processing power to perform the task over a data set of size s: $\mathrm{EPC_i(t_a, s)} < \mathrm{PC_i(t)}$
2) a node $\mathrm{d_i}$ must have enough energy to perform the task over a data set of size s: $\mathrm{EEC_i(t_a, s)} < \mathrm{RE_i(t)}$
3) a node $\mathrm{d_i}$ must have enough memory to perform the task over a data set of size s: $\mathrm{EMC_i(t_a, s)} < \mathrm{M_i(t)}$

During the local assignment phase, a cluster-head, or the set of neighboring cluster-heads in case of the global assignment, will choose the local node, among the ones satisfying the above constraints, that will prolong the life of the corresponding local group by using the following objective function:

$$\mathrm{RL_{LG_j}(t)} = \mathrm{Max} \sum_{\mathrm{i=1}}^{\mathrm{N_{LG_j}}} \alpha_\mathrm{i} \mathrm{RL_i(t)} \tag{11}$$

where $\mathrm{RL_{LG_j}}$ denotes the residual life of local group $\mathrm{LG_j}$, $\mathrm{N_{LG_j}}$ is the number of nodes within the local group $\mathrm{LG_j}$, $\mathrm{RL_i}$ is the residual life of node i in the group, and parameter α_i takes into account the importance of node i in the local group. The node associated with the maximum value in the objective function will be selected by the cluster-head as candidate node. Note that throughout the experimental evaluation the parameter α_i is set to 1 thus, all the nodes have the same role within the local group.

If the global assignment phase is activated, the final decision is taken by considering all the candidate nodes proposed by the neighboring clusters. The task will be assigned to the local group that maximizes the life of the whole network:

$$\mathrm{RL_{net}(t)} = \mathrm{Max} \sum_{\mathrm{j=1}}^{\mathrm{N}} \alpha_\mathrm{j} \mathrm{RL_{LG_j}(t)} \tag{12}$$

where N is the number of groups in the network.

The choice of pursuing first a local optimum is motivated by the intent of reducing the transmission costs. As indicated by several researches, the wireless communication is a major source of energy dissipation in MANETs. We have had this confirmed by the experimental evaluation of the proposed task allocation strategy. In the original design of the algorithm, the global assignment phase is activated when the residual life of the local group is lower than the 50% of its initial value. We evaluated the algorithm over a prototype (a complete description of the experimental evaluation can be found in Section V-A). We found that the communication energy highly degrades the algorithm performance. On the basis of this result we set the threshold to 20% and limited the frequency with which the nodes update their resources state to the corresponding cluster-head. Figure 2 shows the average dissipated energy, separately identifying the communication (ET) and computation terms (EC), for both the old and new version of the algorithm. In particular, the curve ET_OLD represents the communication energy of the original version of the algorithm, while ET_NEW stands for the communication energy of the new version. Obviously, the computation energy component is the same for both versions. From the evaluation is evident the considerable improvement to the algorithm: Figure 2 shows that in the new version of the algorithm the energy for communication is almost negligible compared to the one for computation; conversely, in the original version the energy for communication represents the dominant cost.

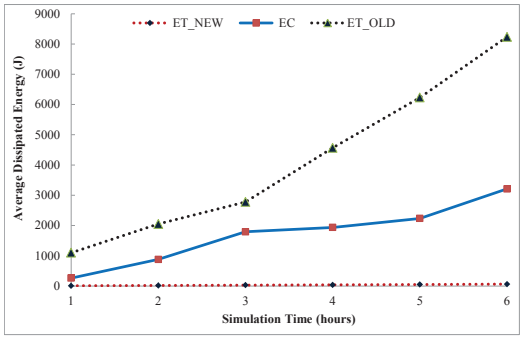

Fig. 2. Average dissipated energy for two versions of the algorithm.

C. The Energy-Aware Task Allocation Algorithm

This section describes the algorithm performed by node when a task has to be executed.

When a node, referred to as requesting node, wants to execute a task it will ask its cluster-head to handle the task assignment process (see Figure 3).

Upon receiving a task allocation request, the cluster-head triggers, through the task allocation method (see Figure 4), the activation of the task allocation strategy allowing to find the optimal allocation for that task. To this aim, the cluster-head verifies whether the residual life of its group is greater than the 20% of its peak value. If the check is successful, it starts up the local assignment phase. If the check is negative or the local assignment failed because none of the nodes within the cluster can execute the task, the cluster-head activates the global assignment phase.

```
Method allocateTaskReq
Input: task t, dataset size s
begin
    if (this.isClusterHead()) then
        allocateTask(t,s);
    else
        myClusterHead.allocateTaskReq(t,s);
    end
end
```

Fig. 3. Task allocation request.

```
Method allocateTask
Input: task t, dataset size s
begin
    localAssignmentFailed ← false;
    if (RL_LG_i > 20%) then
        < d_candidate, ERL_LG_i > ← localAssignment(t,s);
        if (< d_candidate, ERL_LG_i > ≠ ∅) then
            if (d_candidate = this) then
                startTask(t,s);
            else
                d_candidate.executeTask(t,s);
            end
        else
            localAssignmentFailed ← true;
        end
    end
    if (RL_LG_i ≤ 20% or localAssigmentFailed) then
        C ← determineNeighboringClusters();
        globalAssignment(t, s, C);
    end
end
```

Fig. 4. Task allocation by the cluster-head of local group $\mathrm{LG_i}$.

The scheduling starts with the local assignment phase where the cluster-head tries to assign the task to a node within the group that maximizes the cluster residual life (see Figure 5). In this way, the cluster-head determines a *candidate* node in the group meeting the requirements to execute the task in terms of energy, processing and memory constraints. The local assignment phase can be executed either by the cluster-head of the requesting node as well as by the cluster-heads of neighboring clusters involved in a global assignment.

The global assignment phase (Figure 6) also considers the neighboring clusters distant at most three hops from the requesting cluster-head. This restriction is due to the assumption that collaborations are established only among neighboring groups and also to limit the communication costs in the network. Among the neighboring groups will be selected only the ones having a residual life greater than a given threshold (set to 30%), in order to avoid overloading of the local group. After that, in each of such local groups, included the contacting local group, is established a candidate node that best performs the task. Finally, among all the candidate nodes, the task will be assigned to the one that allows to extend the residual life of the whole network.

V. Performance Evaluation

In this section we present an experimental evaluation of the proposed energy-aware scheduler. We have implemented a prototype of the system by realizing a network of mobile devices composed of a set of smart phones and Android emulators.

```
Method localAssignment
Input: task t, dataset size s
Output: candidate node d_candidate, estimated residual life ERL_LG_i of the
local group LG_i if the task would be assigned to d_candidate
begin
    if (this = CH_i or (this ≠ CH_i and RL_LG_i > 30%)) then
        RL_LG_curr ← 0;
        ERL_LG_i ← 0;
        d_candidate ← ∅;
        foreach node d_i ∈ LG_i do
            if (d_i.hasSkill(t,s)) then
                RL_i ← d_i.estimateNodeResidualLife(t,s);
                RL_LG_curr ←estimateGroupResidualLife(d_i,RL_i);
                if (RL_LG_curr > ERL_LG_i) then
                    ERL_LG_i ← RL_LG_curr;
                    d_candidate ← d_i;
                end
            end
        end
    end
    return < d_candidate, ERL_LG_i >;
end
```

Fig. 5. Local assignment of task t within local group $\mathrm{LG_i}$.

```
Method globalAssignment
Input: task t, dataset size s, set of neighboring clusters C
begin
    D ←∅;
    foreach local group LG_i ∈ C do
        < d_candidate, RL_LG_i > ← CH_i.localAssignment(t,s);
        D.add(< d_candidate, RL_LG_i >);
    end
    ERL_net ← 0;
    d_best ← ∅;
    foreach node d_i ∈ D do
        RL_net ←estimateNetworkResidualLife(d_i,RL_LG_i);
        if (RL_net > ERL_net) then
            ERL_net ← RL_net;
            d_best ← d_i;
        end
    end
    CH_best ← getClusterHead(d_best);
    if (d_best = this) then
        startTask(t,s);
    else
        if (CH_best = this) then
            d_best.executeTask(t,s);
        else
            CH_best.requireTaskExecution(d_best, t, s);
        end
    end
end
```

Fig. 6. Global assignment of task t.

Unless otherwise specified, the parameters used in the simulation are as follows. Tasks are generated on each node. The task arrival event is a Poisson process with variable frequency. The simulation area was set to 250000 m^2. The initial value of the transmission range was set to 100 meters and the initial energy level on each device is 24300 J. Each device was equipped with a network interface 802.11 b/g, with a bandwidth of 11 Mbps. The transmission energy is based on the model presented in Section III.

The scheduler model proposed is general and the considered computational tasks may concern different application

scenarios. To the aim of the performance evaluation, we focused on data mining algorithms. In all the experiments, we refer to the computational load as the energy consumption, a priori estimated, to execute a set of data mining tasks. These costs have been estimated through experimental evaluation. We characterize the performance of a data mining algorithm running over a specific device and with respect to a given data set, in terms of the following metrics: memory consumption, battery depletion and execution time assuming that the data mining tasks are CPU-bound.

A. Evaluation Results

Since the goal of our scheduling policy is to maximize the energy levels of the nodes and, thus, extend the network lifetime, the simulation aims to study the behavior of the scheduler with respect to the energy depletion and network lifetime. Accordingly, we use as performance metrics the *residual life of the network*, the *number of devices powered* and the *number of completed tasks*. To assess the effectiveness of the proposed scheduler energy-aware (EA) we compared its performance with that achieved by the well known round robin (RR) scheduling algorithm. This comparison aimed at evaluating the throughput, in terms of number of completed tasks, and the energy consumption of the two algorithms. Note that the comparison through the residual life parameter is significant only if the two algorithms execute the same number of tasks. Otherwise, the comparison can be made with respect to the number of alive devices and the number of completed tasks.

In a first experiment, the initial energy level of the devices in the network is on average the 75% of the peak value. The computational task used is the J48 data mining algorithm running over a data set of 800 Kbytes that takes 2 hours and 16 minutes to be executed, with an average energy consumption of about 2592 J. The total number of submitted tasks is 70; these tasks arrive to the system following a Poisson distribution with a frequency λ equal to 20 tasks per hour. The simulation time is not fixed a priori, since the experiment ends when all the scheduled tasks have been executed. The aim of this first experiment is to show that the proposed EA scheduler is effective in prolonging network lifetime compared to the RR algorithm. In particular, by executing the same number of tasks, the EA scheduler saves about 3 hours compared to RR (see Figure 7). The EA scheduler allocates only 55 tasks of the 70 submitted because recognizes that it is the maximum number of tasks that the devices in the network can execute. The EA scheduler completes 54 of such tasks because for the last allocated task the scheduled node, at execution time, has no longer the energy required to run it. Such node asks for the re-allocation of the task but no other nodes in the network have the energy needed to execute it. Thus, the task execution fails. The RR algorithm executes also 54 tasks, however, it is not aware of the energy availability of the nodes and, thus, allocates a number of tasks greater than the actual availability of the network. Consequently, it causes the turning off of 5 devices (see Figure 8). We can conclude that the proposed

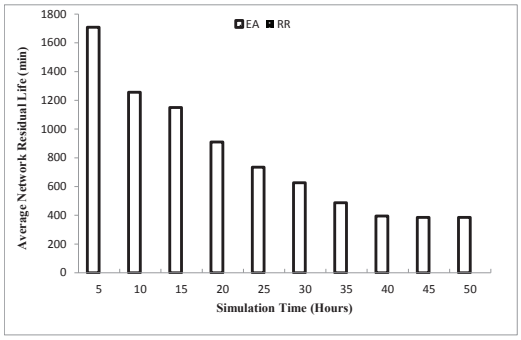

Fig. 7. Average network residual life for both EA and RR algorithms.

EA algorithm allocates and completes the execution of all the tasks that can be executed over the network, prolonging the network lifetime by balancing the energy load and avoiding in such way devices turning off.

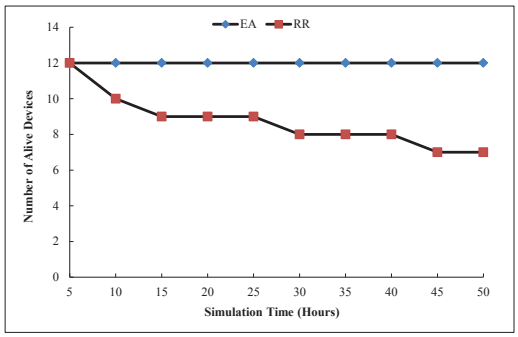

Fig. 8. Number of alive devices after the execution of 54 tasks for both EA and RR algorithms.

In a second experiment the devices are less charged, with an average initial energy available of 30% of the peak value. The simulation time is not fixed, the experiment ends when all the scheduled tasks are executed; by fixing the number of submitted tasks to 15, this time is of 12 hours. Figure 9 shows the number of alive devices and the number of completed tasks for both the EA and RR algorithms with respect to the simulation time. From the graph one can see that EA balances the energy load and remains with all the devices alive whereas RR assigns tasks also to device that do not have the necessary energy and, thus, causes the switching off of 6 devices. Moreover, at the end of the simulation time the EA algorithm completes 15 tasks versus 9 of RR. This means that when the tasks are compute demanding, compared to the overall network energy, the EA algorithm makes the best from the available energy outperforming RR also in terms of number of completed tasks. We repeated the experiment by considering an increasing number of submitted tasks, ranging from 15 to 30. With the considered network energy configuration, the maximum number of J48 tasks, over a dataset of 800 KBytes, that can be executed are 17. Confirming the previous result, EA does not allocate a number of tasks greater than the maximum number supported by the network. It, thus, successfully executes all the allocated tasks. Conversely, RR allocates all the submitted

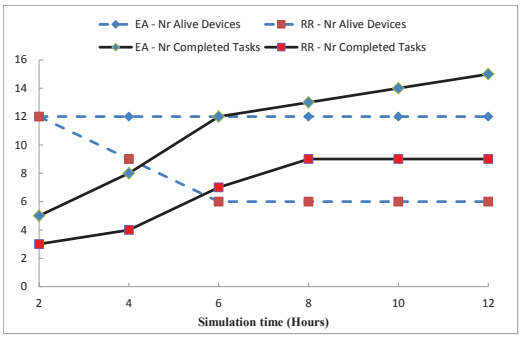

Fig. 9. Number of powered devices and number of completed tasks for boththe EA and RR algorithms.

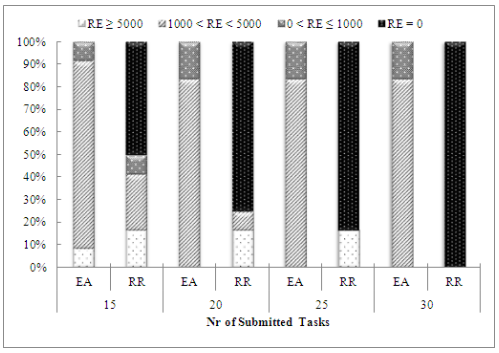

Fig. 11. Distribution of devices remaining energy for both EA and RR algorithms w.r.t the number of submitted tasks.

tasks and, as shown in Figure 10, the number of devices turned on decreases by increasing the number of submitted tasks, reaching zero when 30 tasks are submitted. In a scenario like the one considered, where the energy configuration of the different devices is rather heterogeneous, the energy load balancing effect of the EA is particularly significant. Figure 11 shows how the remaining energy is distributed among all the devices in the network. In particular, it is shown how this distribution changes by increasing the number of submitted tasks, for both the EA and RR algorithms. More precisely, it is specified the percentage of devices with (i) no energy ($\mathrm{RE} = 0$), (ii) low remaining energy ($0 < \mathrm{RE} \leq 1000$) (iii) medium remaining energy ($1000 < \mathrm{RE} < 5000$) and (iv) high remaining energy ($\mathrm{RE} \geq 5000$). One can note the effectiveness of the EA algorithm in balancing the energy level among the devices. In the case of the EA algorithm the percentage of devices with medium remaining energy is always rather high, regardless of the number of submitted tasks. Conversely, in the case of the RR algorithm, by increasing the number of submitted tasks the percentage of devices without energy grows always more, reaching the 100% when 30 tasks have been submitted and all the devices are turned off.

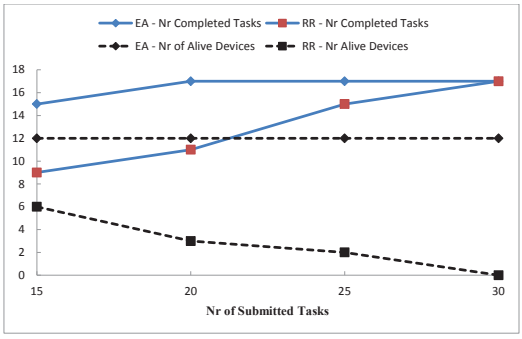

Fig. 10. Number of alive device and number of completed tasks for both EA and RR algorithms w.r.t. the number of submitted tasks.

The efficiency of the EA scheduler is further confirmed when dealing with tasks of increasing computational load. The graph in Figure 12 shows that EA outperforms RR in terms of throughput and device lifetime. By increasing the computational load the EA scheduler completes a lower number of tasks because it allocates only the tasks that can be executed avoiding this way the switching off of the devices. Conversely, RR turns off a number of devices that increases with the growing of the computational load. With a computational load of 2500 J, the two algorithms execute the same number of tasks but EA remains with all the devices switched on while RR turns off more than the 90% of the devices.

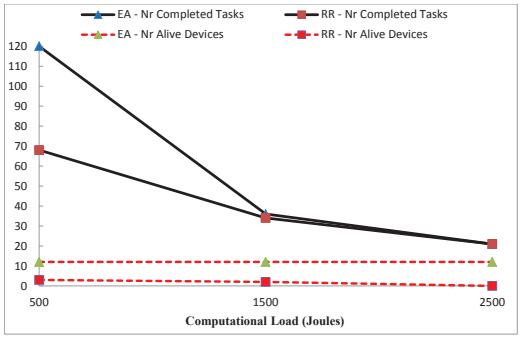

Fig. 12. Number of alive devices and number of completed tasks for both the EA and RR algorithms w.r.t. the computational load.

In the last experiment four different network configurations are considered. For all the configurations the average initial energy level of the devices is the 30% of the peak value. The configurations differentiate only for the distribution of the energy among the devices. The EA scheduler with its dynamic strategy adapts its behavior to the different configurations always assigning the tasks to the most energy powerful devices. Differently, RR allocates the tasks without taking into account the energy availability of the devices. Thus, the performance of RR depends on the specific combination of devices and tasks: each time a task has to be allocated, RR efficacy depends on the charge of the device at the beginning of the scheduling queue and on the load of the task to be allocated. For example, if the most charged device is at the beginning of the scheduling queue when a high computational task has to be allocated, there is a chance of executing the task without turning off the device. Figure 13 shows that EA successfully executes all the submitted tasks in all the considered configurations (as EA performs in the same way

in all the configurations, in Figure 13 we use the notation EA_[1,2,3,4] to refer to EA in the 4 different configurations). The graph also shows the bad behavior of the RR: in all the considered configurations the throughput of EA is higher than the RR one. This is particularly evident in configuration RR_4 where the less charged devices are at the beginning of the scheduling queue. Consequently, a greater number of tasks have been allocated to such less energy powerful devices that are not able to execute them; for example when 5 tasks have been submitted, RR is unable to complete even a single task in correspondence of configuration RR_4.

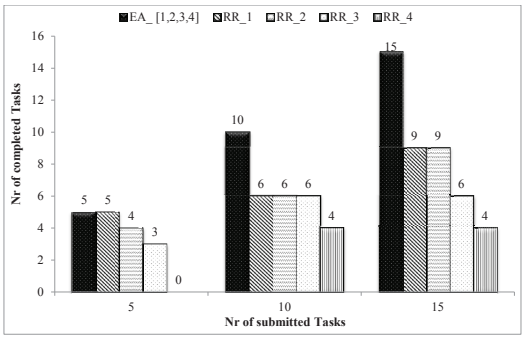

Fig. 13. Number of completed tasks in different energy configurations for both EA and RR algorithms w.r.t. the number of submitted tasks.

In general, the behavior of the RR scheduler which assigns tasks to each processors in equal portions and in circular order, lets its effectiveness depending on the combination task to be allocated and device at the beginning of the circular queue. For example, in the case of Figures 10 and 12 the curves representing the RR performance, are relative to the most convenient configuration for RR, the one where the most energy powerful devices are at the beginning of the circular queue. It is, thus, the configuration which allows to achieve the best performance with RR. Despite this, EA is much more efficient. If in those experiments we had considered one of the other possible configurations for RR, the improvement of our algorithm would have been even higher.

VI. Related Work

Most of the existing research work in the area of energy-aware systems are hardware-based techniques focusing on reducing the energy consumption of the processor. One of the most adopted techniques is turning off idle components [5]. Dynamic Voltage Scaling (DVS) is another technique of energy conservation. DVS refers to the technique of simultaneously varying the processor voltage and frequency as per the energy performance level required by the tasks [6], [7], [8]. Remote execution is a software-based technique in which a device with limited energy transfers a computational task to a nearby device which is more energy powerful.

Energy-aware task scheduling is another software method where the scheduling policy aims at optimizing the energy. To the best of our knowledge, little work has been done on energy-aware scheduling over a MANET network. In [9] is proposed an energy-aware dynamic task allocation algorithm over MANETs. However, this work is different from ours in terms of the underlying architecture and cost function to be optimized. We clustered the devices to promote local cooperation among nearby devices and minimize the transmission energy. This issue is particularly relevant because we have experimentally found that the transmission energy highly impacts on the overall energy consumption. In contrast to ours, the solution proposed in [9] is effective for compute intensive applications and does not address the communication aspects of the system. Furthermore, we adopt a different objective function: we maximize the network residual life rather than minimizing the energy consumption. Using the residual life parameter we are able to actually consider the real energy consumption rate of single devices, single clusters and the overall network. Conversely, [9] does consider only the local computation issues and works at a node level ignoring the workload in the rest of the network. Thus, differently from us, they do not take into account the actual load of the devices with the possibility of assigning a task to a device that consumes less energy, but which is less charged compared to another one that consumes more energy but it is more energy powerful and thus could efficiently execute that task.

VII. Conclusion

In this paper we presented a task allocation scheme for mobile networks focusing on energy efficiency. To conservatively consume energy and maximize network lifetime we have introduced a heuristic algorithm that balances the energy load among all the devices in the network. We have implemented a prototype of the system and evaluated the scheduling strategy through simulation experiments. Results show that the proposed scheduler greatly enhances the performance of the system compared to time-based traditional schedulers like the round-robin. We achieved improvements in terms of network lifetime, number of alive devices and number of completed tasks.

References

[1] C. Comito, D. Talia, and P. Trunfio. "An Energy-Aware Clustering Scheme for Mobile Applications'. *IEEE Scalcom'11*, pp. 15–22, (2011).

[2] R. Garey, D. Johnson. "Complexity bounds for multiprocessor scheduling with resource constraints". *SIAM J. Computing*, 4:187–200, (1975).

[3] L. M. Feeney. "An energy-consumption model for performance analysis of routing protocols for mobile ad hoc networks". *Mobile Networks and Applications Journal*, 6(3):239-250, (2001).

[4] H. W. D. Chang, W. J. B. Oldham. "Dynamic task allocation models for large distributed computing systems". *TPDS*. 6:1301–1315, (1995).

[5] K. Li, R. Kumpf, P. Horton and T. Anderson. "A Quantitative Analysis of Disk Driver Power Management in Portable Computers". *USENIX* conference, pp. 279-292, (1994).

[6] J. Zhuo, C. Chakrabarti. "An efficient dynamic task scheduling algorithm for battery powered DVS systems". *ASP-DAC'05*, pp. 846–849, (2005).

[7] Y. Zhang, X. Hu and D. Chen. "Task scheduling and voltage selection for energy minimization". *DAC'02*, pp. 183–188, (2002).

[8] H. Aydin, R. Melhem, D. Moss, P. Mejia-Alvarez. "Power-Aware Scheduling for Periodic Real-Time Tasks". *IEEE Transaction on Computers*, 53(5):584–600, (2004).

[9] W. Alsalih, S. G. Akl, H. S. Hassanein. "Energy-Aware Task Scheduling: Towards Enabling Mobile Computing over MANETs". *IPDPS'05*, pp. 242a, (2005).