Integrating Demand Response Capability into Smart Appliances: Design and Evaluation of Distributed Scheduling

Alessandro Parisi [a], Ken Christensen [b]

[a] DIEI (Dipartimento di Ingegneria Elettronica e dell'Informazione), Universiy of Perugia, Italy

[b] Department of Computer Science and Engineering, University of South Florida, USA

Abstract

The need for a constant balance between power demand and supply, together with the lack of cost-effective solutions for storing electric energy, affects the efficiency of power grids and limits the integration of renewable sources. Dynamic pricing and demand response programs provide mechanisms to regulate the power demand according to supply conditions. We focus on designing an autonomous scheduler for appliances and electric vehicle charging stations. The proposed system enables the integration of automatic demand side management functionalities. This paper presents the system architecture, the required communication services, the scheduling problem, our scheduling algorithm, and a simulation-based performance evaluation. We believe that combining dynamic pricing and peak capping represents a viable solution to effectively achieve peak shift and manage real-time events such as forced outages. Our scheduler is based on a distributed collaborative architecture. It processes appliance tasks characterized by a deadline and a load profile. We propose an offline heuristic method that first reorders and then allocates all tasks by using alternative scheduling heuristics. We compared our offline algorithm with a brute force online cost minimization algorithm and an optimal greedy scheduler. Experimental results show how performance varies with workload and how the rescheduling mechanism significantly increases the number of schedulable tasks.

Introduction

Electric power must be consumed in the moment it is produced. Dealing with this constraint is critical for energy service providers and grid operators, whose primary objective is avoiding shortages in electricity generation. Since electricity demand fluctuates and cost-effective solutions for buffering electric energy are not available, generation systems normally employ operating reserves to ensure power grid reliability during peak consumption hours. The generation capacity is thus under-utilized and the overall power system efficiency drops. Another consequence of this scenario is the limited percentage of renewable sources that can be integrated into the grid, due to their intermittent nature. Peak consumption thus affects the efficiency of power systems and the environmental impact of energy production.

Having the ability to manage the electricity demand, in order to shift loads to off-peak hours, could mitigate the problems related to the balance of demand and supply of electricity. In this context, demand response (DR) programs provide mechanisms to regulate the power demand through curtailment signals according to conditions of the supply side, whereas dynamic pricing tariffs offer retail energy prices that vary by following the wholesale cost of electricity. According to a 2008 study by Capgemini, DR programs could achieve 25-50% of the EU’s 2020 targets concerning greenhouse gas emission reduction and energy efficiency improvement [1].

We believe that power use in residential buildings can be scheduled by local controllers relying on a distributed communication protocol to achieve peak shift and to react to the state of the power grid. To this aim, we propose a smart device scheduler (SDS) that controls the operation of smart appliances and electric vehicle (EV) charging stations, based on energy service provider signals.

We highlight the importance of considering methods to manage the electric consumption for residential buildings. In 2007, the residential sector accounted for the 37.0% of the total electricity consumed in U.S. [2] and for the 28.2% of that consumed in EU-27 [3]. A 2008 Smart-A project study also shows how, in a typical European house, a washing machine, a tumble dryer, and a dishwasher together account for a significant average consumption of 597 kWh per year and how those appliances can be shifted up to 9 hours [4]. Also, the growth of the EV market may have a major impact on power systems. This concern is demonstrated by the efforts of the EDISON project that addresses the integration of charging systems into the future smart grid through a vehicle-to-grid framework [5].

In the actual electricity market, it is crucial to regulate the demand according to the marginal costs of production [6]. In an ideal scenario where the demand follows price variation with enough elasticity, dynamic pricing strategies are sufficient to achieve that goal. Unfortunately, electricity consumers are not accustomed to dealing with varying prices, so their responsiveness is low. Regarding this, Braithwait states that the "responsiveness depends on price levels, customer types, and the degree of technology assistance provided" [7]. Also, Nordman suggests that the adoption of dynamic pricing is necessary but not sufficient to guarantee the needed level of dynamism for load consumption adjustment [8]. The idea behind our SDS is to improve the level of responsiveness by using peak cap signals together with a dynamic pricing tariff. By peak cap we intend a DR program wherein users accept terms under which a fine will incur if their consumption exceeds the established cap in a given time frame. In this way, occurrences such as wrong wind forecasts and grid outages can be translated into real time events (price and/or peak cap changes) that will be communicated by the energy service provider and locally managed by each SDS.

Aside from economic benefits, users could be encouraged to subscribe to DR programs if supported by automated energy management systems (EMS) that minimize costs and provide feedback about energy consumption and charges. Also, in our vision, EMSs should operate autonomously according to grid information signals and users should be able to override the EMS choice, in order to ultimately decide when and how much energy to consume. Finally, EMSs should scale with the number of controlled devices, in order to extend the managed domain to a multi-premise scenario. The proposed SDS, presented in the next section of this paper, deals with those requirements by relying on a distributed logic implemented locally in each controlled smart device. Schmidt and Van Laerhoven define smart devices as "devices that are not ignorant about their environment and context" [9]. In the following, we will limit the meaning of the term to the interface towards other smart devices, the smart grid (signals from the energy service providers), and local meters (consumption signals). More in detail, this paper does not address topics related to home automation, such as the processing of signals from presence or thermal sensors, in order to predict consumption and determine device operation. Also, the paper does not address DR economic aspects, nor does it address the potential acceptance of the proposed DR based SDS by consumers, grid operators, and service energy providers. The contributions of this paper are:

- The design of a distributed framework for the energy management of residential buildings based on energy service provider signals.
- The formulation of the scheduling problem for controlling the operation of smart devices, in order to minimize energy costs under power cap and deadline constraints.
- The simulation-based performance evaluation of an algorithm based on the reordering of requests and online scheduling heuristics.

The third section of this paper presents the scheduling problem. Each local scheduler processes the tasks of the smart device in which it is deployed. This is done by an online cost minimization algorithm with deadlines and power cap constraints. If this algorithm fails to schedule a task, all of the local schedulers in the domain collaborate to reschedule the previously allocated but unexecuted tasks. The rescheduling mechanism might also run when real-time events impose new rates or caps. This represents an offline scheduling problem which, to the best of our knowledge, there exists no solution. We propose an offline heuristic method that first reorders and then allocates all tasks by using alternative scheduling heuristics. The method has been implemented in C language and the simulation-based evaluation results are described in the fifth section of the paper.

Energy management

With the term energy management we refer to the process of controlling the operation of smart devices by scheduling their start (and rescheduling when necessary), in order to satisfy the deadline set by the user, comply with DR power caps, and minimize energy costs according to a dynamic pricing tariff.

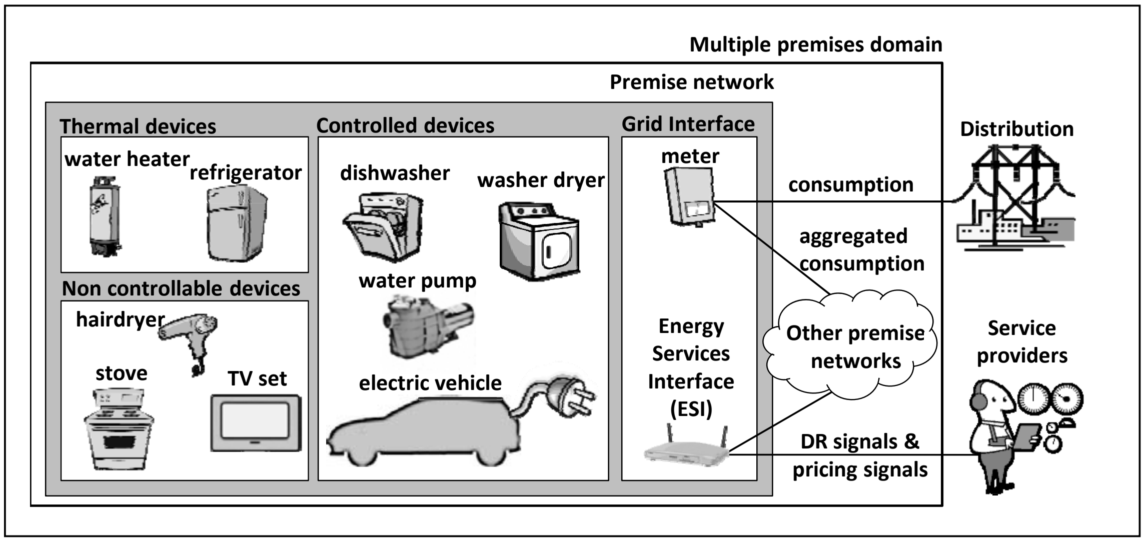


Figure 1 – Reference scenario: components and interfaces of the multiple premises domain controlled by the smart device scheduler (SDS).

To enable energy management and meet the needs of users, grid operators, and energy service providers described in the previous section, we identify the following requirements for the SDS:

1. The SDS should operate in an autonomous way, according to the information obtained from the energy service provider. No direct control signals from the grid should be employed.
2. The SDS should take into account the electric consumption of the non-controlled loads of the controlled domain.
3. The SDS should be distributed in order to scale with the number of controlled devices. The presence of a central controller should not be required.
4. The SDS should manage real time events signaled by the grid. These events may require the rescheduling of already scheduled tasks, as well the interruption of running tasks, in order to adjust the local consumption to the current state of the grid.

In the presented work, we do not address the interruption of running tasks. Also, we consider only those smart appliances with an operation cycle that may be shifted within a time frame of hours, such as dishwashers. We do not address the scheduling of thermal based appliances, such as refrigerators and air conditioners, which are characterized by a periodic operation cycle. The reason is twofold: the potential time shift of these appliances is limited [4]; the duty cycle of those devices is a function of the ambient and target temperatures, thus, in order to effectively manage such appliances, our SDS should integrate home automation functions capable of controlling the target temperature in response to the ambient temperature and the consumer's preferences, as proposed by Le May al. [10], [11]. We assume that the user interface logic has the responsibility of issuing scheduling requests. The operation of such a functional block is out of the scope of this paper and we assume that requests cannot be changed by our system. Finally, we do not consider the integration of policies to fairly distribute the energy among the managed loads. Nevertheless, we believe that a fairness mechanism is needed to properly manage the aggregated consumption of a multi-premises domain.

In the following we use the term timetable as a local repository where the SDS nodes store the information regarding energy price, peak limits, the consumption of non-controlled devices, and the schedule of other controlled devices. The timetable will be further described in the next section.

Real time events may refer to emergency or market situations that are translated into SDS requests to modify the timetable. For instance, events may refer to emergency situations that might impose the drop of the peak cap levels in a very short time frame in order to preserve the reliability of the grid. With the term "real time" we mean that those events must be managed by the SDS as soon as they are received: they may or may not require the rescheduling of tasks.

The reference scenario is presented in Figure 1. The SDS architecture is distributed, so the same logic is implemented in each controlled device (dishwashers, washing machines, dryers, washer dryer

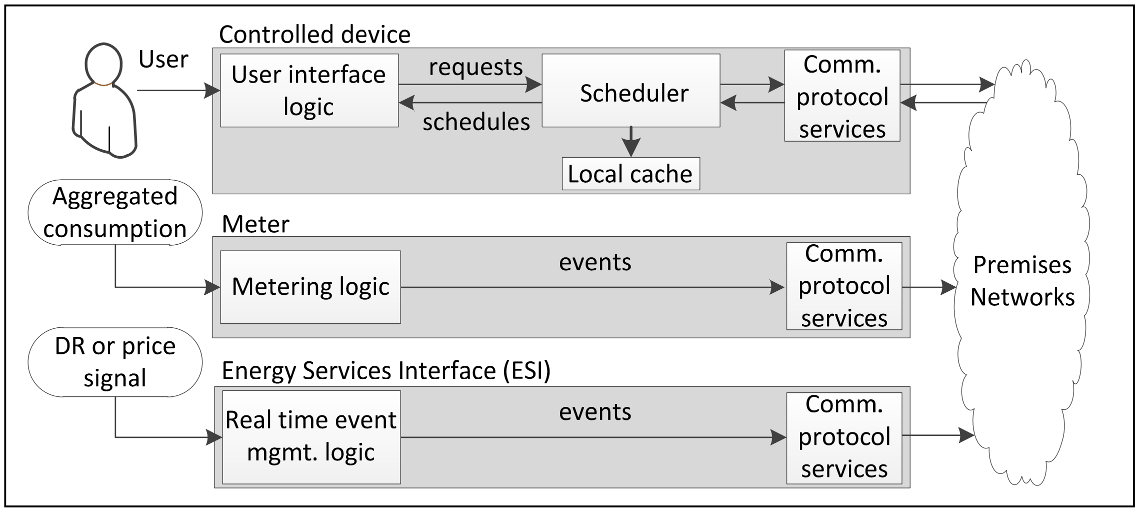


Figure 2 – Functional blocks of the proposed smart device scheduler (SDS).

combos, water pumps, defrost and ice making functions of refrigerators, and EV charging stations). We assume that all components of the premise domain communicate through the domain network. In case of a multi-premise domain, the network includes all of the residential units. The premise interface towards the grid follows the smart grid reference diagram proposed by NIST [12]. It includes two logical blocks: the meter, through which energy usage is communicated, and the Energy Services Interface (ESI), through which the SDS interacts with energy service providers and receives DR (peak cap) and dynamic pricing signals as a consequence of real time events that take place in the grid. The ESI is in charge of relaying this information to the other domain nodes.

The consumption of non-controllable devices stored in the timetable is given by a baseline forecast provided by the meter. In case the effective consumption differs from the baseline by a given threshold, the meter is in charge of communicating the new baseline by broadcasting it within the domain network.

In the case of a multi-premises domain, we assume coordination mechanisms among the ESIs of each premise, in order to relay the grid signals only once, as well as among the domain meters, so that the baseline is evaluated only once on the basis of the aggregate consumption.

Figure 2 presents the functional blocks of the SDS. We have included the user interface logic for controlled devices, the metering logic, and the real time event management logic in order to define the interface of our system. We have introduced the functionalities provided by these blocks above, but their operation is out of the scope of this work. The core block of the SDS is the scheduler. To comply with the requested distributed architecture, we propose a distributed collaborative scheduler. It is collaborative since the local scheduling logic of each node interacts with the other nodes in order to retrieve information regarding the overall domain schedule and to send rescheduling commands. This is done by relying on the services provided by the communication protocol. The timetable is stored in a local cache. Figure 2 also shows how the meter and the ESI broadcast events to the domain network on the basis of aggregated consumption and grid signaling, respectively.

We assume that the peak consumption is evaluated periodically, for instance, in intervals of 15 minutes. In other words, the instantaneous power draw can exceed the peak cap, whereas the constraint is relative to the average value during a particular time period. So we divide the time axis into slots, each having a given energy limit.

The communications protocol must provide services that enable the following actions.

- Scheduling: the scheduling node needs to retrieve the schedules of all other nodes; the ESI and meter need to communicate real time events to all domain nodes.
- Rescheduling: it has to be ensured that only one node can schedule tasks at time; the scheduling node needs to communicate commands to change the schedule of the other nodes.

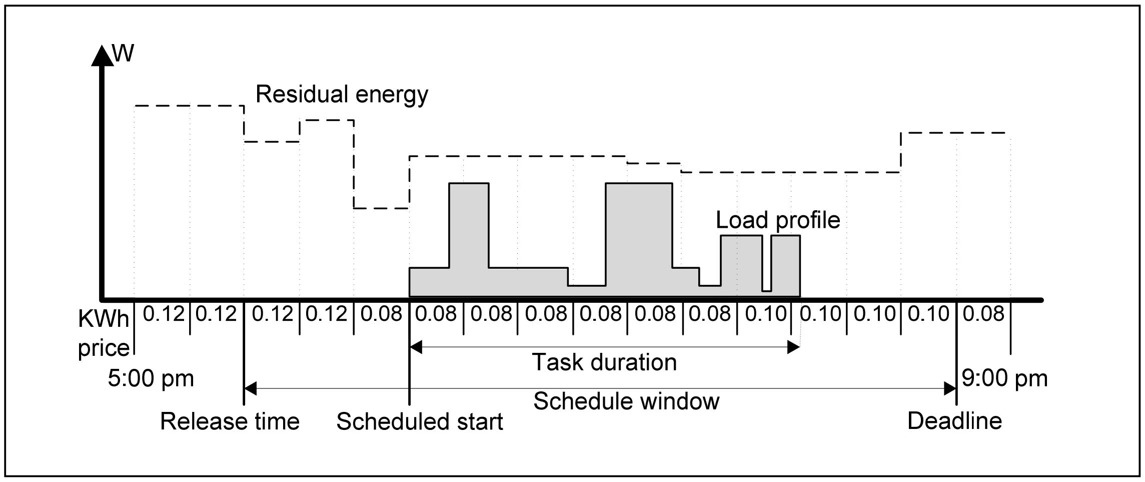


Figure 3 – Timetable with energy prices and residual energy values for each time slot of 15 min, in which an example of appliance task scheduling is reported.

Scheduling

By following the SDS requirements, we identify those for the scheduler functional block:

1. The scheduler should allocate tasks in time by minimizing costs (energy price) and complying with temporal (release time and deadline) and resource (residual energy) constraints.
2. The scheduler should reschedule tasks, in order to react to real time events when necessary.
3. The scheduler should be distributed and each device should implement the same logic.

Scheduling problem

Task requests arrive one at time, as an online problem. Thus we address the problem of allocating the current request. By referring to the timetable presented in Figure 3 we introduce the scheduling terminology.

A task is a process that a smart device needs to perform. We consider appliance and EV recharging tasks. Appliance tasks are characterized by a load profile having a fixed duration (for instance, a task of 120 min that requires 0.2kW for the first 80 min and 1 kW for the remaining 40 min). The duration of a task is expressed in number of timeslots. In Figure 3, a typical dishwasher load profile is used [13]. EV recharging tasks differ from those of appliances in that they require the scheduling of a given amount of energy. The rate at which the energy is provided may vary, but it must comply with a range of admissible power values imposed by the charging technology. So, EV recharging tasks have no fixed duration and that degree of liberty can be exploited by the scheduler.

Each timeslot is characterized by a rate (cost per kWh) and a cap (for example, a power cap of 4 kW for the average consumption that corresponds, in the case of a timeslot of 15 min, to an energy cap of 1 kWh). We define residual energy as the available energy for the scheduling of the current task. For each timeslot, the residual energy is obtained by subctracting the non-controlled consumption forecast and scheduled consumption from the energy cap. In the Figure 3 timetable, the residual energy of each timeslot is given by the area of the rectangle subtended by the time axis and the dashed line.

Tasks must be scheduled to start after a release time and end before a deadlline. Those values are expressed as a timeslot index. We introduce the following definitions:

- The *schedule window* is the time interval [release time, deadline – 1].
- The shift interval is the time interval [release time, deadline – duration].
- The shift time is a duration value given by: *time shift = deadline – duration – release time + 1.*

We assume that tasks start at the beginning of the scheduled timeslot. Also, we assume that the number of timeslots of the local cache timetable is large enough to comply with the task deadline. On the other hand, there is no guarantee on the availability of residual energy.

Online scheduler

For appliance tasks, we can visualize the scheduling problem as a two dimensional online problem, where a load profile figure must be placed within the schedule window in such a way that it does not exceed the residual energy dashed line and minimizes the energy price, as in Figure 3. We propose a brute force approach. Our algorithm consists of finding, within the shift interval, a set of candidate timeslots. Each candidate has a number of consecutive timeslots with enough residual energy to host the load profile. For each candidate timeslot, the energy cost for the entire task allocation is determined. Finally, the algorithm chooses the minimum cost candidate as the scheduled start.

As for an EV request, we adopt a valley filling approach that consists of allocating the task where there is available energy in the timetable. At every iteration step, the algorithm chooses the minimum cost timeslot within the schedule window. For each chosen timeslot, the amount of allocated energy must comply with the range imposed by the charging technology. Also, the remaining residual energy must be higher than a pre-configured parameter, in order to avoid saturating the timetable.

Rescheduling: offline heuristic

In the case a request turns out to be unfeasible, our idea is to reschedule the previously allocated tasks to achieve a better utilization of the residual energy. More in detail, if the online cost minimization algorithm fails, the scheduling node will reallocate all of the previously scheduled but unexecuted tasks of the controlled domain as well as the current task. This represents an offline problem. The rescheduling mechanism might also take place when a real time event imposes changes in the timetable. In this case, the SDS verifies if the overall schedules comply with the new residual energy timetable and if the total allocation cost is the minimum achievable, according to the new price timetable. To the best of our knowledge, the offline problem described above has no solution available in literature. We propose a method based on heuristics. Our method exploits the knowledge of the entire set of requests, even though it ultimately relies on an online scheduling algorithm (with the term “online” we refer to an algorithm that processes one task at time). Our strategy consists of reordering the tasks before scheduling them. Our intuition arises from the observation that by reordering tasks we influence the scheduling outcome. We deduce from the scheduling problem characteristics that:

- If the scheduler allocates the most flexible requests first (characterized by a relatively long shift interval), it may result in emptying those timeslots that are critical for the scheduling of later tasks with more restrictive requirements.
- Different load profiles may be more or less easy to schedule. Thus, choosing to schedule the less regular profiles first may be convenient.

Figure 4 shows two simple non-realistic scheduling cases that demonstrate how the reordering may succeed. In such examples we consider time slots of 1 hour. Presented in (a) are the load profiles of three task requests of the first example. They are characterized by the same [4pm, 9pm] schedule window, that corresponds to the interval of the timetable presented in (b) that also shows energy costs and residual energy values of 5 kWh and 3 kWh. The online scheduler first processes the tasks in the arrival order: first A, indicated with the couple (A,1), then B and finally C. In (b) it is showed how there is no space for B or C, since the algorithm has scheduled A in the slots where the cost is lower. Thus the timetable utilization is not optimal and this is indicated by the exclamation mark. In (c), the same online algorithm achieves a complete feasible schedule after reordering the tasks as (B,1), (C,2), and (A,3). In (d) the four task requests of the second example are presented. They have the same schedule window [4pm, 6pm]. As in the previous example, (e) shows how the scheduler fails in allocating the tasks by following the arrival order. In particular, there is no space left for the D request. In (f) is presented the output of the successful scheduling after reordering the requests as (A,1), (C,2), (B,3), and (D,4).

The proposed reordering heuristic consists of assigning a priority value to each task. Tasks are then sorted in decreasing order of priorities. The task priority is a weighted sum of three different combinations of task parameters that are related to how flexible the request is and to how easy it is to find a feasible schedule. For each task, we define the task priority parameter as follows:

$$priority = x \cdot energy + y \cdot shift + z \cdot shape. \qquad (1)$$

For a given task, *(x, y, z)* are weights for, respectively, the total energy requested *(energy)*, the shift time *(shift)*, and the product of the task duration and the maximum energy requested in a timeslot *(shape)*. The *energy* and *shift* values are normalized by the relative mean values over the *N* requested tasks of the offline problem.

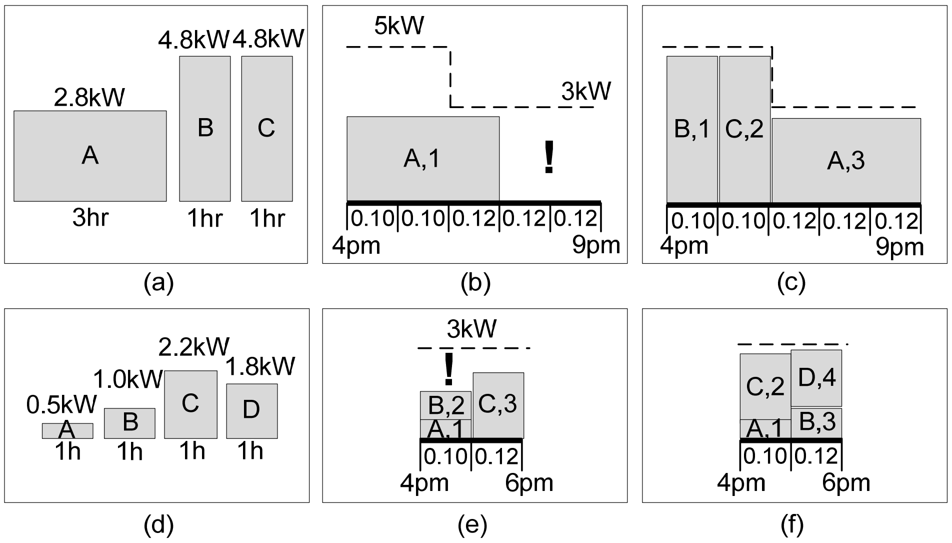


Figure 4 – Two simple non-realistic scheduling cases: motivation for the reordering approach.

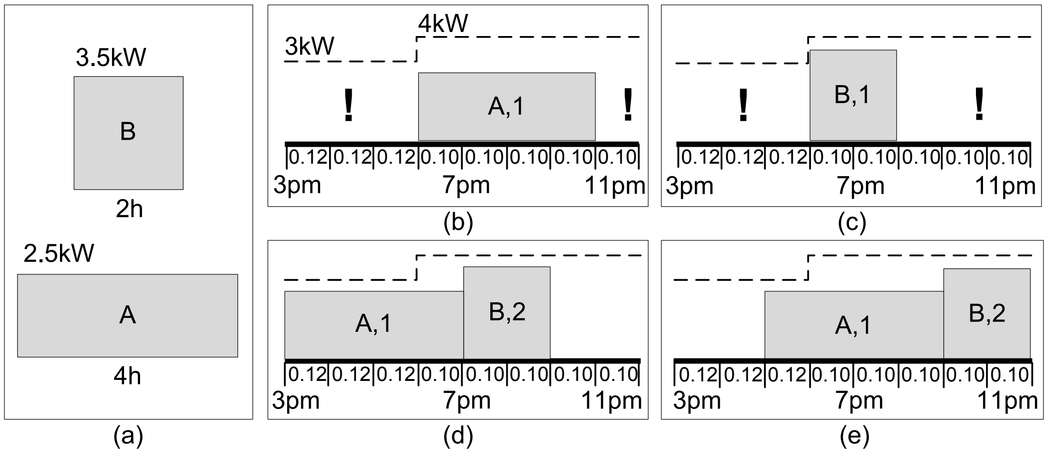


Figure 5 – A simple non-realistic scheduling case: motivation for different scheduling strategies.

There are cases in which a feasible schedule for each of the *N* tasks of the offline problem exists, but no task reordering allows achieving it. Figure 5 reports a simple non-realistic example of this. As in the previous cases, we consider time slots of 1 hour. Two requests are presented in (a), A and B, that have the same schedule window: [3pm, 11pm]. Presented in (b) is the timetable as well as the outcome of the online algorithm (which just tries to minimize the cost) after processing the A task: there is no space left for B. It should be noted how the algorithm uses the first available slot that minimizes the cost (to say, it picks the 6pm one instead of 7pm). This is reasonable since the SDS should try to minimize the delay annoyance. In (c) we have the output of the offline reordering approach. The requests are reordered so that the tallest task, B, is processed first by the online algorithm. As shown, after allocating B, there is no space left for A either. In (d) we have the schedule given by a modified online algorithm. It processes the tasks by their arrival order. Besides aiming at cost minimization, this algorithm tries to reduce the timetable fragmentation by determining a schedule that minimizes the resulting residual energy. This is done by considering a cost tolerance of 0.10 cost units. That cost tolerance sets a constraint to quantify the trade-off between cost minimization and fragmentation reduction. As to say, “find an alternative schedule that reduces the resulting residual energy, but do not spend more than 0.10 compared to the minimal cost schedule”. By choosing the 3pm slot for the task A, the algorithm is spending 0.06 cost units more, but the resulting residual energy allows the scheduling of the task B. Finally in (e) we have the optimal solution, achieved by the same “minimal residual energy” online algorithm used in (d), but with a cost tolerance of 0.3 cost units.

The situation presented in the previous example motivates the use of two scheduling heuristics in alternative to the brute force cost minimization algorithm. They trade the cost minimization objective for a better allocation of the residual energy timetable. By following the previous example, for a given task request, those heuristics first obtain the schedule candidates and the minimum cost schedule, in the same way as the cost minimization algorithm does. They then consider the set of schedule candidates whose energy charges differ from the minimum one by less than a given cost tolerance. Among those candidates, a timeslot is chosen as the final scheduled start by following two different criteria: Max Min Residual Energy and Min Total Residual Energy.

The first scheduling criterion is such that the chosen schedule is the one that shows the maximum minimum residual energy value. This value is determined over the timeslots necessary to allocate the load profile. The minimum residual value $m_i\,[\,j\,]$ of the timeslot *j* for the scheduling of task *i* is:

$$m_i[\mathrm{j}] \stackrel{\mathrm{def}}{=} \min_{k\in[j,j+l_i-1]} \{\, R[\mathrm{k}] - e_i[\mathrm{k}-\mathrm{j}+1]\,\}. \tag{2}$$

Where l_i is the duration of task *i* expressed in number of timeslots, *R[k]* is the residual energy of the timeslot *k*, and $e_i\,[\,j\,]$ is the energy requested by the task *i* in the timeslot *j* of its load profile. If we indicate the minimum cost by *c*, the cost of allocating the task *i* in the timeslot *j* by $c_i\,[\,j\,]$, and the cost tolerance by *p*, we can define the set of schedule candidates S_i of the task *i* as:

$$j \in S_i \qquad j \text{ is a schedule candidate and } c_i[\mathrm{j}] - c < p\,. \tag{3}$$

The criterion thus imposes to choose as the scheduled start the timeslot s_i given by:

$$s_i = \mathrm{j}: \quad \max_{\mathrm{j}\in S_i} \{\, m_i[\mathrm{j}]\,\}. \tag{4}$$

We refer to this scheduling method as Max Min Residual Energy (MMRE). If we consider the scheduling problem in a graphical way (as a two dimensional problem), by adopting this criterion, the scheduler avoids fitting the figure where the space is little. On the contrary, this method tries to allocate the load profile shape where the space left for future requests is bigger. Following this criterion may result in choosing timeslots with greater residual energy or, more interestingly, timeslots in which the particular energy profile finds a better fit. In this way, the resulting timetable will more likely be able to host future tasks characterized by a long duration.

The second scheduling criterion goes in the opposite direction. It requires choosing as the scheduled start the timeslot that has the minimum total residual energy value. We define the total residual energy of the timeslot *j* for the scheduling of the task *i* as:

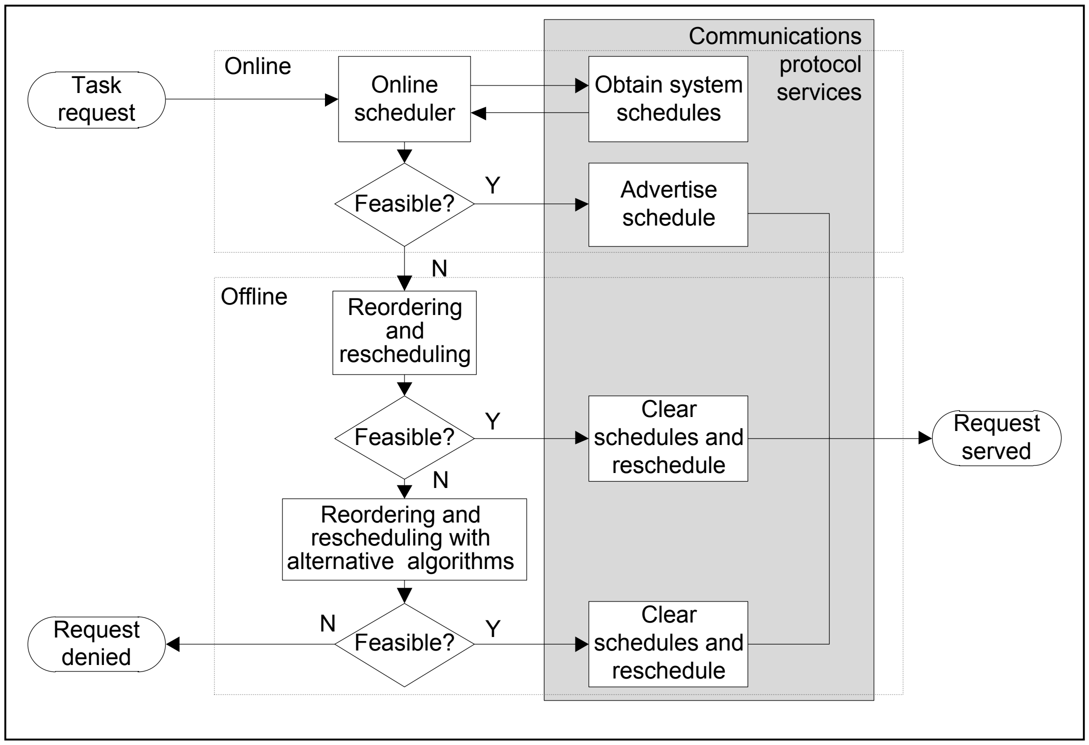
$$w_i[\mathrm{j}] \stackrel{\mathrm{def}}{=} \sum_{k=j}^{j+l_i-1} R[\mathrm{k}]. \tag{5}$$

So, the schedule is obtained by:

$$s_i = \mathrm{j}: \quad \min_{\mathrm{j}\in S_i} \{\, w_i[\mathrm{j}]\,\}. \tag{6}$$

In this way, by using the same graphical visualization of the scheduling problem, the scheduler tries to allocate the load profile shape where the available space is just enough to host it, in order to obtain a better packing. The scheduler does not care about the remaining space for future figures to be allocated. So this method is more suited for cases in which the task requests are characterized by a large shift interval. We refer to the heuristic based on this criterion as the Min Total Residual Energy (MTRE) since it tries to minimize the residual energy left.

To summarize, for the offline problem of scheduling *N* tasks, we propose a heuristic method that consists of reordering the task requests and schedule each of them by using an online algorithm. Besides the online cost minimization algorithm, we propose three scheduling heuristics: 1) MMRE, 2) MTRE, 3) RND. The last one consists in randomly choosing, for each on the *N* tasks, one method between MmRE and mTRE, with equal probability.



Communications protocol services
Online
Task request
Online scheduler
Obtain system schedules
Feasible?
Y
Advertise schedule
N
Offline
Reordering and rescheduling
Feasible?
Y
Clear schedules and reschedule
Request served
N
Reordering and rescheduling with alternative algorithms
Request denied
N
Feasible?
Y
Clear schedules and reschedule

Figure 6 - Distributed scheduling: process flow and communication services.

In Figure 6, we illustrate the process flow of the scheduler and the required communication services. This picture places the brute force cost minimization algorithm (represented by the online scheduler block) together with the offline method based on the reordering and alternative online scheduler heuristics. The online scheduler receives the task request from the user interface logic block. Then it advertises the start of the scheduling process in the SDS domain, in order to avoid potential scheduling conflicts, and retrieves the set of system schedules from all other smart devices to update the timetable. If the scheduling process succeeds, the scheduler advertises its new schedule to the other devices by broadcasting a message. The schedule is also communicated to the user interface logic block. In case the schedule is not feasible, the offline method reschedules all previously allocated tasks. First, the tasks are reordered and scheduled with the cost minimization algorithm (the same one used in the online part). If this method also fails, the scheduler will employ alternative online algorithms (MMRE, MTRE, or RND). If the one of the offline approach succeeds, the scheduler will rely on the communications protocol to manage the rescheduling process that will require broadcasting a control message to let the other devices reset their own schedule timetable. After that, unicast messages will be employed to communicate the new schedules to the rescheduled devices. Finally, if the offline method does not succeed in scheduling all tasks, the current request is denied, otherwise the schedule outcome is communicated to the user.

Performance evaluation

We choose to simulate the SDS since an analytical model of the offline scheduling problem is not available. The objective of the performance evaluation is to verify the efficacy of the rescheduling-based method that executes whenever the online algorithm fails in scheduling a task. In particular, the proposed method must reschedule all the previously scheduled tasks plus the current one (offline problem). If at least one of those tasks is not scheduled, the current request is denied. As a consequence, we evaluate the ability of the rescheduling method to entirely schedule a set of tasks.

The simulated system consists of the scheduler block presented in Figure 2. The reference scenario is a multi-premises domain. We assume that the timetable information is completely available. We have implemented the scheduler block by following the process flow presented in Figure 6. Also, in separate tools, we have implemented the online brute force scheduler (which is a part of the scheduler block), an optimal offline scheduler based on a greedy iterative tree search algorithm, and a workload generator. All of the tools have been developed in C language.

Table 1 - Comparison between scheduler methods: percentage of sets entirely scheduled.

Interarrival rate (λ)	Feasible sets	Rescheduling method	Online scheduler
0.4	76.2%	90.1%	14.5%
0.6	66.9%	87.2%	12.2%
0.8	62.2%	86.1%	11.6%
1.0	56.5%	85.4%	11.3%
1.2	54.1%	84.7%	11.1%
1.4	52.3%	84.1%	11.0%
1.6	51.2%	83.4%	10.8%

The input of the three scheduler tools consists of four different files containing the energy prices, the power caps, the non-controlled consumption, and the task requests, respectively.

The workload generator provides stochastic sets of task requests. The tasks are related to different realistic load profiles of the following appliances: dishwasher, tumble dryer, washing machine, washer dryer, pool pump, well pump, dehumidifier, and the automatic defrost mechanism of refrigerators. We do not consider EV charging tasks since the valley filling algorithm has a major impact on the residual energy and, consequently, it becomes very difficult to analyze the scheduler performance. For each set of tasks, the workload generator creates a CSV file containing requests characterized by a load profile (represented by a duration value and a data structure indicating the power request over time), a release time, and a deadline. The temporal values are expressed in number of timeslots. The processing follows a bottom-up approach. For each appliance load profile, a bounded Pareto RNG determines the relative number of requests and, for each of these requests, an exponential RNG determines an interarrival value and a deadline delta. The release time of each task is determined by accumulating the interarrival values of the previously processed tasks. The deadline is then obtained as the sum of the release time, the load profile duration, and the deadline delta.

In the presented experiment, we compare the outcome of the scheduler block, the online scheduler, and the optimal scheduler. The adopted metric is the percentage of entirely allocated sets of tasks. The workload is made of 5000 sets of 12 tasks, so the workload generator considered only the first 12 tasks of each set. In this case, the choice to use such a low number of tasks is due to the factorial computational complexity of the greedy scheduler algorithm that would not execute in a reasonable time with larger sets of tasks. The rate parameter of the exponential distribution of interarrivals (indicated by λ) is the experimental factor, in order to verify the influence of the workload. Increasing λ results in smaller interarrivals, so the schedule windows of different requests will more likely overlap and the number of sets that cannot be entirely scheduled will increase. The timetable time range is 48 hours, divided into timeslots of 15 min. As for the energy prices, we employ a Time of Day tariff that consists of four periods of 6 hours over a day, with prices equal to 0.6, 0.8, 1.2, and 1.4 cost units, respectively. Power caps and the non-controlled consumption are chosen in such a way to bring the timetable close to the saturation, according to the workload. Regarding the parameters of the rescheduling method, the *x*, *y*, and *z* weights of the priority reordering formula presented in Equation (1) are chosen as 2, 3, and -1, respectively. Those values were obtained by a previous experiment for the tuning of the system. The cost tolerance *p* presented in Equation (3) is set to 5 cost units.

The results are presented in Table 1. The results of each experiment run are reported in different rows. The first column presents the λ factor used to obtain the 5000 task request sets of each run. The second column reports the outcome of the optimal scheduler. That value corresponds to the percentage of feasible sets of tasks. It is evident how this value drops when the workload becomes heavier. The third and fourth columns give the percentage of feasible sets that are successfully scheduled by the proposed rescheduling method and the online algorithm, respectively, in order to estimate how close the two algorithms get to the optimal solution. In the first row, for a workload with interarrivals with λ = 0.4, the percentage of feasible sets of tasks is 76.2%. So, among the 5000 sets of requested tasks, 3810 sets are completely schedulable. Of these 3810 sets, the online scheduler is capable of entirely allocating only the 14.5%, whereas the system scheduler allocates the 89.5%. With heavier workloads, the performance of the two scheduler methods drops, as shown in the other rows of the table. We can conclude how the rescheduling mechanism significantly increases the number of schedulable tasks when compared to the online approach (brute force cost minimization algorithm). Also, we can affirm that the scheduler performance varies significantly with the workload.

Related work

The following works address demand side management (DSM) and scheduling techniques. The integration of building automation technologies for energy efficiency and DSM have been addressed in [10] with a centralized architecture where the main component is the Unified Hub that receives utility price signals from the smart meter. The signals are then relayed to appliances that may control themselves or delegate the DR controls to the hub. The same research group proposes an energy management system based on the previous architecture that has the objective of maximizing the user comfort while complying with energy consumption constraints [11]. The system is based on a blackboard architectural pattern and includes appliance usage detectors, sensors, and other components. A collaborative recommender is in charge of selecting the suitable building control algorithms for being installed in the blackboard. Our work does not address home automation functions. We focus on improving the electricity demand responsiveness by peak cap DR signals.

The integration of smart appliances into the smart grid is addressed by the Whirlpool Smart Device Network (WSDN) technology [13]. WSDN consists of a complex DR architecture that includes both the house and smart grid domains. Three levels of DR operation are proposed: (1) appliances individually respond to signals; (2) the operation of all smart devices is coordinated by a home energy management system; (3) the Smart Grid coordinates the DR of the houses through the Internet. The Authors state that consumers should ultimately control how appliances response to the DR signals. The main difference with our architecture is in the fact that appliances are managed by a centralized Smart Device Controller. The paper also provides typical load profiles of several appliances.

A scheduling problem for tasks characterized by deadlines and a given power demand is presented in [14]. The objectives are peak shaving (peak shift in a short time frame) and cost minimization. The reference DR program is based on local control through real time pricing signals. The scheduling is analytically formulated as a min-max problem (NP-hard, since it is a particular case of a well-known problem). The proposed algorithm is based on the Longest Process Time greedy search algorithm, but processes the task with the largest energy consumption first according to the cumulative cost of each timeslot. Through an analytical formulation, the Authors prove that the algorithm finds near optimal solutions. The benefits of energy storage are also considered. We did not provide an analytical formulation of our scheduling problem, which is more complex since we also consider power peak constraints. We use a similar a max-min approach for the proposed MMRE heuristic.

A DSM distributed scheduler for domestic energy consumption (smart appliances and EVs) is presented in [15]. The reference scenario is a time of use DR program that controls the aggregated consumption of multiple houses. The scheduler implemented in each Smart Meter and the local algorithm relies on the knowledge of the current overall schedule of the domain. The Authors present a formulation of the problem based on Game Theory. The distributed algorithm aims at reducing the total energy cost and the peak to average ratio. Thanks to the analytical formulation, it is shown that the system reaches the global optimum when the local algorithm reaches the local optimum. The proposed distributed architecture is similar to our collaborative scheduler. Also, we consider a multiple house scenario. Nevertheless, the Authors do not consider peak caps in their scheduling problem.

The scheduling of periodic appliances (such as refrigerators and air conditioners) through the control of their duty cycles is addressed in [16]. The proposed algorithm (Tetris-like) sets the phase of the tasks related to a common time period in order to achieve peak shaving. The Authors present the problem of how to reduce the cooling of multiple air conditioners in a fair manner and propose a max-min approach based on temperature. In our work, we address the scheduling of tasks that can be shifted in a time frame of hours, by minimizing cost and complying with deadlines and peak caps.

Summary and future work

This article presents a smart device scheduler that enables peak shift in residential buildings by controlling the operation of smart appliances and electric vehicle charging stations. Our system tackles the problem of improving the level of responsiveness of domestic electricity demand by offering an autonomous rescheduling mechanism that reacts in real time to dynamic pricing and peak cap based demand response signals. The scheduler is distributed and collaborative and does not require a central controller, thanks to the services provided by a communications protocol. We formulate the offline scheduling problem related to the rescheduling process, characterized by deadline and power peak constraints. To the best of our knowledge, there is no solution available in literature. We propose and evaluate a heuristic-based method. The presented results, obtained by simulating the system, show how the rescheduling method increases the number of successfully allocated tasks when compared to a system that only uses an online scheduler. In particular, the proposed method achieves a successful schedule of up to 90% of the sets of tasks that are shown to be entirely schedulable by an optimal scheduler (greedy), whereas an online scheduler is capable of scheduling up to 15% of the same sets. Future research work consists of 1) defining a mechanism for the interruption of running tasks, 2) addressing the need of a fairness mechanism to allocate energy among multiple houses, 3) formalizing the specifications of the communications protocol, and 4) building a prototype of the smart device scheduler. We also believe that the definition of an analytical model of the offline problem could be an important research contribution, which would enable a study regarding the existence of an optimal solution for at least a subset of the original problem.

Acknowledgements

We would like to thank our anonymous reviewers who have provided helpful comments for improving the paper.

References

[1] A. Chardon, O. Almén, P.E. Lewis, J. Stromback, B. Château, “Demand Response: a decisive breakthrough for Europe. How Europe could save Gigawatts, Billions of Euros and millions of tons of CO2”, Capgemini consulting, 2008.

[2] “Annual Energy Outlook 2010 – with projections to 2035”, U.S. Energy Information Administration, April 2010.

[3] “Final electricity consumption by sector (ENER 018)”, European Environment Agency, September 2010.

[4] R. Stamminger, “Synergy Potential of Smart Appliances”, SMART-A Deliverable 2.3, 2008.

[5] EDISON project website http://www.edison-net.dk.

[6] H. Fraser, “The importance of an active demand side in the electricity industry”, The Electricity Journal, Elsevier, Volume 14, Issue 9, pp. 52-73, November 2001.

[7] S. Braithwait, “Behavior modification”, Power and Energy Magazine, IEEE, Vol. 8, Issue 3, pp. 36-45, May-June 2010.

[8] B. Nordman “Beyond the Smart Grid: Building Networks”, Environmental Energy Technologies Division - Lawrence Berkeley National Laboratory, May 2010.

[9] A. Schmidt and K. Van Laerhoven, “How to build smart appliances?”, IEEE Personal Communications, Vol. 1070-9916/01, pp. 66-71, August 2001.

[10] M. LeMay, R. Nelli, G. Gross, and C. A. Gunter, “An integrated architecture for demand response communications and control”, Proceedings of the 41st Annual IEEE Hawaii International Conference on System Sciences (HICSS '08), Waikola, Hawaii, January 2008.

[11] M. LeMay, J. J. Haas, and C.A. Gunter, “Collaborative recommender systems for building automation”, Proceedings of the 42nd Annual IEEE Hawaii International Conference on System Sciences (HICSS '09), Waikola, Hawaii, January 2009.

[12] “NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 1.0”, NIST Special Publication, NISR, January 2010.

[13] T. J. Lui, W. Stirling, H. O. Marcy, “Get Smart”, Power and Energy Magazine, IEEE, Volume 8, Issue 3, pp. 66-78, May 2010.

[14] J. Xiao, J. Y. Chung, J. Li, R. Boutaba, J W. Hong, “Near Optimal Demand-Side Energy Management Under Real-time Demand-Response Pricing”, POSTECH University, Pohang, South Korea, 2010.

[15] A-H. Mohsenian-Rad, V.W.S. Wong, J. Jatskevich, R. Schober, “Optimal and Autonomous Incentive-based Energy Consumption Scheduling Algorithm for Smart Grid”, Proceedings of the IEEE Conference on Innovative Smart Grid Technologies, January 2010.

[16] J. Howard, H. Ham, N. F. Maxemchuk, “Smart Air Conditioners”, Proceedings of the Global Smart Grid Symposium, US-Korea Conference on Science, Technology, and Entrepreneurship (UKC), August 2010.