# Flexible Load Shedding Using Gossip Communication in a Multi-agents System

Victor Lequay, Mathieu Lefort, Saber Mansour, Salima Hassas

# Flexible Load Shedding using Gossip Communication in a Multi-Agents System

Victor Lequay*†, Mathieu Lefort*, Saber Mansour†, Salima Hassas*
*Université Lyon 1, LIRIS, UMR5205, F-69622, France
Email: {victor.lequay,mathieu.lefort,salima.hassas}@liris.cnrs.fr
†Ubiant SA, Lyon, France
Email: saber.mansour@ubiant.com

***Abstract*—In order to balance production and consumption on the power grid, efforts are mostly made on the production side. Demand response is a classical solution consisting in harnessing electricity consumption to allow the introduction of renewable energy sources and self-production in the energy mix. In this context, the main problem is to coordinate the network nodes, considered here as agents, in order for them to be able to anticipate their need and also to adjust their contribution to the collective load-shedding effort. To tackle this last point we present an original bottom-up approach of distributed load-shedding with a decentralized algorithm based on gossip protocols. These protocols offer both reliability and scalability to be used on a large scale power grid. On this ground, we built an efficient decentralized control mechanism using a self-evaluation process improving the system performances. In this paper, we present our model and evaluate it using realistic simulated data. We then discuss current limitations and further improvements as part of future work.**

## 1. Introduction

The energy sector undergoes dramatic changes as home electricity consumption raises steadily. Indeed, to satisfy the energy demand, new production units have to be plugged, adding complexity to the network as well as financial and environmental costs. Moreover these equipments are designed to sustain peaks in demand, which means that they are almost never operating at full capacity, thus generating additional costs. A simple solution to this problem would be to act on the demand side to spread or reduce the load peaks instead of increasing production to match them (see figure 1). This concept, named load shedding, allows to greatly reduce the environmental impact of electricity production while being way cheaper. Load shedding has been around for many years but was almost uniquely used in the industrial sector because consumption is controllable and highly demanding operations can be anticipated days ahead and easily postponed if necessary. A broader use of load shedding in network management needs to extend its principles to smaller, less predictable and distributed loads : homes. This idea, named distributed load shedding, may seem obvious at first glance, but a fair number of constraints must be satisfied to be integrated in the main power grid and accepted by the end user. This means that in addition to reactivity, reliability and capacity criteria, the system has to be strongly ergonomic and adaptive.

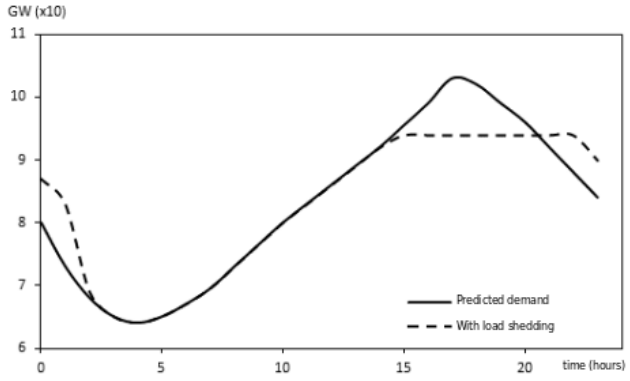



Figure 1: Example of the effect of load shedding on a demand peak.

In this article, we present a solution which satisfies all these constraints. Our work is based on Hemis[1], a smart building energy management system developed by Ubiant in collaboration with the LIRIS laboratory. Hemis is able to compute in real-time the load shedding capacity of a building with respect to the user's comfort. This information will be transmitted to the system we present in this article, whose decentralized architecture allows to precisely and adaptively control and shed the energy consumption of a large number of buildings. This architecture has been designed with scalability in mind, paving the way to manage thousands or even millions of homes with minimal need for an increasing processing power or any costly network infrastructure. We had two main goals. Firstly, to be able to operate in near real-time while maintaining the high quality of service needed for a smooth integration in the power grid. Secondly, to make our system as light as possible, allowing us to manage a possibly large amount of homes. These two competing objectives led us toward a fully decentralized approach that we explain here.
We will first present an overview of the related work on

1. http://www.ubiant.com/hemis-ubiant/

distributed load shedding in section 2. We then introduce our model in section 3 and discuss our simulation results in section 4, before concluding and presenting some possible ways to improve our model as part of future work.

## 2. Distributed load shedding

Load shedding techniques have been used for long to balance consumption in the power grid, but until now it was mostly deployed to big consumers like factories or shopping centres. The adjective distributed refers to the integration of smaller loads into the system. In this paper, load shedding will always refer to distributed load shedding as we designed our system to allow any kind of load of any curtailment capacity. Load shedding raises many issues at different levels of abstraction, that could be summarized in 3 main points differentiating the existing approaches.
The first one is the transmission media chosen to exchange orders and information between the operator and the end users. It represents an important part of the system costs and plays a major role on its scalability and reliability. A direct link permanently established between each client and the operator means a heavy and costly infrastructure in most cases, but also allows a nice reactivity. On the contrary, if connections are not permanent, the frequency of the connection becomes critical to the quality of the shedding. This is why most of the works on this topic use a permanent connection to the operator.
The second main issue is the kind of control the system has over the loads to start a shedding. Usually, a price signal is used : the provider changes the energy price to encourage users to consume more or less, depending on the needs of the grid. This kind of incentive system is extremely easy to set up but suffers from bad reliability because the end user's reactivity to a change in electricity price is never guaranteed and even less predictable [2]. Moreover, if the end user fails to manage his energy consumption correctly and continues to use his appliances normally when the price goes up, the impact on his electricity bill could be enough for him to withdraw completely from the system. These issues make this price based approach more adapted to less restrictive needs, like the global curtailing on daily peaks, where simple pricing schedules can be issued to consumers. Nevertheless, as a signal or incentive, the electricity price can be an interesting ground for building distributed load shedding systems [9][7]. More recently, the interest of the energy sector has been caught by more straightforward controlling solutions.

The third point is the algorithm that controls the connected loads. The challenge is to provide a quick and reliable answer on a given curtailment capacity, then maintain the stability of this curtailment until the end of the event, while making sure that the user's comfort is not altered and ensuring as much as possible a fair sharing of the effort between each concerned load. On this topic, existing approaches can be classified according to the degree of centralization of the decision process. On one end of the spectrum, a central server sends simple binary orders to connected appliances which respond immediately. On the other end, each load decides when to switch on and off, sometimes depending on external information.

Centralized algorithms applied to some real cases are mainly based on statistics. Historical systems, like the Centralized Remote Control by Musical Frequency ("Télécommande Centralisée à Fréquence Musicale") from EDF (historical French operator) [3], allow the operator to remotely shut down some equipments like the water heater if the user subscribed to a dedicated contract. Also, as most of the consumption is basically delayed, such a method may later create an even higher peak than the one the operator wanted to reduce at first [2]. This effect is naturally reduced by stochastic systems which randomly spread the loads curtailment and can be avoided in a centralized way only if loads are individually manageable, as in this case it is easy not to power on all appliances at once. More recent works on centralized load shedding systems sometimes use techniques coming from the field of artificial intelligence like artificial neural networks [5] or reinforcement learning [14] to automatically choose the most efficient strategies.

Some decentralized approaches use multi-agents systems. Often based on price incentives [9][7], most of them use a stochastic approach to force spreading of the agent's actions in time, thus smoothing fluctuations or overshoots that could happen when using a simple hysteresis system [12][14]. Likewise, Beal and al. [11][10][1][8] suggest a stochastic algorithm called ColorPower which allows a distributed control of the energy demand. The idea is for the end user to set up a kind of connected plug that can switch off or on his devices. This plug switches randomly with a probability that is constantly updated according to a signal sent by a central server. This signal increases the probability of shutting down during a curtailment effort, and reduces it otherwise. These systems have one major drawback which is due to their probabilistic structure : they cannot be efficient unless a substantial amount of plugs are connected. Indeed, using this system, the available curtailment capacity is not the total one because every load does not automatically take part in the effort. That is why purely stochastic systems need a minimum number of partakers in order to ensure a sufficient load shedding capacity and are hard to set up incrementally [4].

## 3. Our model

To solve the different issues raised by the existing systems seen above in terms of user comfort, cost, adaptability and scalability, we chose to overturn the problem and to start from the user. Our bottom-up approach is to our knowledge the first one to focus In this section, we will start by presenting Hemis, the smart energy management solution over which we will build our system. We will then explain the goals our model has to achieve and also the constraints it has to comply with. We will also describe our decentralized model for a near real-time distributed load-shedding management.

### 3.1. Hemis: Home Energy Management Intelligent System

Developed by Ubiant in collaboration with the LIRIS laboratory, Hemis is a smart building energy management system. Based on a multi-agents system solving a multi-constraints optimisation problem, Hemis is able to maintain a desired level of comfort (meaning a correct temperature, lighting, air quality, etc... with respect to the user's habits and preferences) while automatically lowering the overall energy consumption. Connected appliances (heaters, lights, sensors, electrical roller shutter, etc...) are represented in the system by reactive agents collaborating on shared marking spaces to achieve the objectives of all environmental factors. The user simply sets theses objectives and the agents collaborate to find the best trade-off between comfort and energy consumption.

Hemis relies on two sources of information to determine the curtailment capacity of the building, that we call flexibility. The first one is a learning based on a local record of the user's habits and energy consumption, combined with real-time data from the sensors (if the weather is unusually cold, the electrical consumption will not be the same as the last days). The second one is the sum of individual anticipations from the agents themselves. The predicted flexibility takes into account every connected or known device, even if they are not currently on. Indeed, a load shedding can either be done by shutting down some equipments but also by not turning on an appliance when it was supposed to be, by advancing or postponing its use (for example by heating a room a little earlier while benefiting from the thermal inertia).

It is important to notice that the user stays in control of its equipments even during a load shedding. If an appliance is switched on despite a shedding being engaged, it will not be forced off by the system. The flexibility is then simply updated with this new piece of information.

### 3.2. The problem statement

In the following of the paper, we assume there exists an intelligent system as the one presented in the section 3.1 giving us the ability to know at each time $t$ the flexibility $f^a$ of the building $a$, which is defined as the amount of energy the building can shed without diminishing the user's comfort. The reliability of this data is of course crucial to the efficiency of the system but our architecture is robust enough to tolerate a difference between the anticipated and the real value of the flexibility. Let $N = \{a_1, a_2, ..., a_n\}$ be the set of buildings equipped with this system. At each time, the total flexibility (or total shedding capacity) of the system is $F = \sum_{a \in N} f^a$. To satisfy a request of a given curtailment capacity $Q$, each agent $a$ will reduce its consumption by some amount $x^a$. During the event, it is very likely that one or more buildings will have their flexibility updated, most of the time reduced or even brought down to zero. To ensure a stable curtailment, the system will have to update $x^a, \forall a \in N$ as quickly as possible so that $\sum_{a \in N} x^a = Q$ at almost all time. In the next section, we introduce our architecture that achieves these stability and reactivity requirements in a decentralized manner and without using any kind of price incentive.

### 3.3. Our distributed load shedding architecture

To answer the problems outlined in section 3.2, we propose a decentralized approach based on a multi-agents system inspired by gossiping algorithms. An agent represents a building that knows its flexibility in real time, as presented in section 3.1. Thanks to a gossip-based aggregate computation algorithm, that we will describe in section 3.3.2, each agent also knows at each time the value of various variables concerning the whole population. These variable are essential to the system as they are the core of the coordination between the agents. They will be described in section 3.3.1. The aim of the decentralized system is to determine for each load shedding event the amount of participation of each agent. For this purpose, each agent locally self-evaluates on various criteria and assigns itself a mark, representing its level of performance. When the agent receives a load shedding order, it will engage a certain amount of its total flexibility depending on its mark. Following the event, the agent will update its mark according to its performance. This process allows the system to improve the quality of the load shedding at each event (see section 3.3.3 for details).

**3.3.1. Shared variables.** For the algorithm to work, agents need to know the value of some variables carrying information about or concerning the entire population.

- The order $O$ is a triplet $O = \{Q, t_d, t_f\}$ with $Q$ the amount of energy to shed between $t_d$ and $t_f$. This order is initially sent to one or more agents by the energy producer in a fully reliable way.
- The total participation $agg$ is the sum of all the capacities the agents have committed themselves to be able to shed at every moment.
- The best mark $g_{max}$ is the highest computed mark of any agent at a given time.
- Variable $c_{max}$ corresponds to the highest participation of an agent amongst the population during the $m$ last events.
- Variable $r_{min}$ corresponds to the lowest failure average of an agent amongst the population.

These variables are propagated and computed in a decentralized manner with the Push-Sum algorithm that we describe in the next section.

**3.3.2. The Push-Sum algorithm.** The Push-Sum algorithm, presented by Kempe et al. [6], allows a group of agents to collectively process the value of an aggregate information (sum, mean, product, maximum, minimum value) in a completely decentralized way using what is called an epidemic

or gossip propagation model. In their approach, each agent $a$ possesses three variables : $x^a$ is the agent's value that will be aggregated with the other's (the agent's part in the shedding for instance), $s^a$ is a sum - initialized to $x^a$ - and the weight $w^a$ - initialized to 0 or 1 depending on the kind of calculation. If we want to compute a sum for example, $w$ will be set to 0 except for one agent whose $w$ will be set to 1. At $t = 0$, each agent sends the pair ($s^a$, $w^a$) to itself. At each time step, a fraction $\alpha_{a,j}$ of $s^a$ and $w^a$ is sent to a set of $J$ agents so that $\sum_{j \in J} \alpha_{a,j} = 1$. Values of $s^a$ and $w^a$ are then updated by summing every pair $(\alpha_{j,a}s^j, \alpha_{j,a}w^j)$ received by $a$ : $s^a = \sum_{j \in J} \alpha_{j,a}s^j$ and $w^a = \sum_{j \in J} \alpha_{j,a}w^j$. The estimate value of the aggregate for one agent is $agg^a = \frac{s^a}{w^a}$. Kempe et al. show that, "given a $\gamma$ and a $\delta$, the relative error in the approximation of the real value $agg$ is at most $\gamma$ with probability at least $1-\delta$ in at most $O(\log n + \log \frac{1}{\gamma} + \log \frac{1}{\delta})$ rounds". This logarithmic complexity allows for scalability. The topology of the network defines the choice of the neighbours to which an agent sends its messages. The number of neighbours $J$ depends on the network topology and the connectivity constraints. Convergence speed will increase with the number of contacted agents at each step. This algorithm also only works if the network is a connected graph, meaning that there is a path between each pair of nodes. The nature of the Push-Sum algorithm allows us to add and remove agents in real-time without affecting the stability of the system, the only consequences being a small disturbance in the values of the shared variables as the information propagates and an obvious variation of the total flexibility in the system. This "plug-and-play" property of our model makes it fault tolerant regarding connectivity. In this article, in addition to a decentralized summing, we also need to compute minima and maxima (for the agents to be able to rank themselves amongst the population). Computing minima or maxima consists in simply keeping the minimum or the maximum of the received values and propagating it. In these cases, values are not weighted by $\alpha$.

It is relevant to notice that this algorithm allows for a straightforward implementation of performance monitoring features for the use of an operator. Thus, we will be able to know the number of agents having a specific value for any given parameter (reliability, participation, etc...), or the average participation and other standard statistical tools.

**3.3.3. Self-evaluation.** The agent computes its mark according to three criteria :

- reliability, which is the ability to reduce its consumption of the amount it was committed to shed, and hold its consumption at the same level during the entire event,
- the average flexibility it possesses given by Hemis,
- its frequency of participation to the previous efforts.

Each one of these three criteria is evaluated considering the history of the agent and gives a mark between 0 and 1. The final mark of the agent is computed as a weighted mean of these three values.
More precisely, the history $M^a$ of an agent $a$ is a list of triplets $M_a = \{(r_1^a, c_1^a, t_1^a), ..., (f_m^a, c_m^a, t_m^a)\}$ over the last $m$ events with

- $c_i^a$ the amount of energy shed by the agent $a$ during the event $i$
- $r_i^a$ the standard deviation to the initial commitment of the agent during the event $i$
- $t_i^a$ equals 1 if the agent participated to the event $i$, 0 else

The agent's mark $g^a$ is computed from its reliability $R^a$, its average capacity $C^a$ and its turnover $T^a$, based on the history $M^a$.

- The reliability score of the agent $a$ is given by $R^a = \frac{r_{min}}{R^a_{moy}}$. The value $r_{min} = \min_{a \in N} R^a_{moy}$ is a shared variable (see sections 3.3.1 and 3.3.2) and $R^a_{moy} = \sum_{j=1}^{m} r_j^a$ the sum of the standard deviation to the initial commitment over the last $m$ events. The maximum mark will be obtained by agents having the smallest variations in their participation.
- $C^a_{moy} = \sum_{j=1}^{m} c_j^a$ being the sum of agent's capacity over the last $m$ events, the capacity score of the agent $a$ is $C^a = \frac{C^a_{moy}}{c_{max}}$, while $c_{max} = \max_{a \in N} C^a_{moy}$ a shared variable updated collectively as seen in sections 3.3.1 and 3.3.2. It is basically a global ranking rewarding agents with big shedding capacity.
- The turnover is computed as follow $T^a = 1 - \frac{\sum_{i=1}^{m} t_i^a}{m}$. This mark goes up if the agent does not participate often, increasing its chance to participate to the next events. An agent who participated to all the last $m$ events will obtain $T^a = 0$.

At the end, we have $g^a = k_1F^a + k_2C^a + k_3T^a$ with $k_1$, $k_2$ and $k_3$ being weighting coefficients so that $k_1 + k_2 + k_3 = 1$ and $k_1, k_2, k_3 \in [0, 1]^3$.

These coefficients are continuously and autonomously adjusted by each agent. The reliability coefficient $k_1$ is increased when the curtailment quality decreases, that is when the sum of the difference between the requested load reduction and the effective one over the event is higher than during the previous events. This aims to choose reliability over capacity when there is a poor quality of curtailment. The capacity coefficient $k_2$ increases with the amount of time needed by the population to reach a consensus when an order is received, relative to the requested variation magnitude. Indeed, if the aggregated value is already near the targeted one, the time needed to reach a consensus on this value will be shorter than for a bigger gap. The goal here is to improve the ranking of the agents providing the biggest shedding capacity, in order to speed up the consensus process. These coefficients are set to some default value at the starting of the system, but could potentially be optimised beforehand.

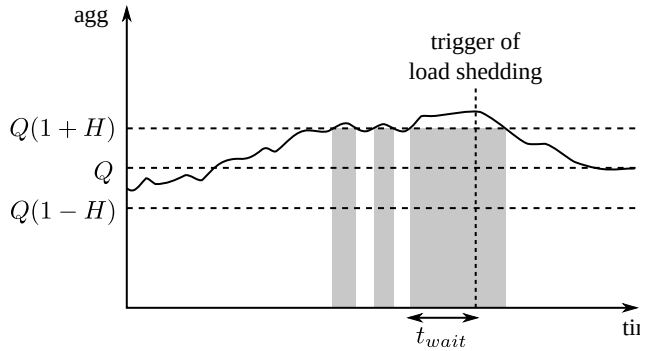



Figure 2: Target crossing detection process. A crossing is confirmed when the aggregate value is above $Q(1+H)$ (grey areas) for a duration of $t_{wait}$. The agent then starts adjusting its share to the effort.

This coefficient adjustment process allows the system to adapt itself to changes in the user's behaviour through time, which could hardly be anticipated as it depends on multiple factors as economical or weather conditions for example.

**3.3.4. Adjusting agent's participation.** Wuhib et al [13] used the Push-Sum algorithm to design a decentralized threshold crossing detection algorithm. Adapted from some of their concepts (an hysteresis and a low-pass filter), we propose a new algorithm that allows each agent to adjust its share on a collective effort to quickly reach a global objective by summing their contributions. The first step is to detect a gap between the estimated sum of the contributions $agg^a$ and the global objective $Q$. It may seem like an easy task, but individual contributions are constantly changing. To avoid an oscillatory behaviour when $agg^a \simeq Q$, two complementary mechanisms are used. The first one is the use of a ratio $0 \leq H \leq 1$ implementing an hysteresis around the target $Q$, damping small variations at low frequency due to the continuous variations of the agent's flexibility (see figure 2). A threshold crossing is then detected if $\left|\frac{agg^a}{Q} - 1\right| \geq H$.

The second mechanism allowing a reliable detection is a kind of low-pass filter erasing quick variations. It is implemented with a timer $t_{wait}$. To be taken into account, a crossing must be observed for at least $t_{wait}$ time steps. These two systems are combined to lead to an accurate threshold crossing detection by each agent despite the frequent variations due to the convergence of the collectively computed aggregate.

When an agent receives a shedding order, it first determines its maximal contribution $x^a_{lim}$ regarding its flexibility $f^a$ and its mark $g^a$ following the formula $x^a_{lim} = \frac{g^a}{g_{max}} f^a$, $g_{max}$ being the highest mark in the population (see section 3.3.1). This way, the agents with the higher marks contribute more than potentially less reliable agents or than those who participated more frequently. If an agent observes a gap between the estimated global effort and the objective as seen earlier, it adjust its own contribution $x^a$ with the formula $x^a = x^a \left(1 + \frac{Q - agg^a}{agg^a}\right)$. If the low-pass filter timer $t_{wait}$ is correctly defined, it allows the agents population to stabilize the new value of the collective effort $agg^a$ before adjusting again, preventing additional swinging that could be due to continuous adjustment of individual contributions. If during its adjustment an agent reaches its virtual contribution limit $x^a_{lim}$, it updates this limit with $x^a_{lim} = \min\left(x^a_{lim} + (f^a_t - x^a_{lim}) * v; f^a_t\right)$, $v$ being a variation coefficient ($v \leq 1$).

## 3.4. Pseudo-code

Algorithm 1 represents the program run by an agent $a$. Variables are initialized during step 1. One will notice that we set $w$ to 0 in this code, which means that one of the other agents will have its $w$ set to 1 for the aggregation computation to work as a sum, as we explained when describing the Push-Sum protocol (section 3.3.2). Apart from this difference, every agent runs the exact same program. Steps 2 to 9 contain the implementation of the Push-Sum protocol.

The processing and the propagation of shedding orders are done in steps 10 through 15. The virtual limit $x^a_{lim}$ is set on step 12.

Steps 16 to 28 correspond to the threshold crossing detection and the adjustment of the agent's contribution. The hysteresis is found at step 16 as well as the low-pass filtering with the timer $t_{wait}$.

Steps 24 to 28 allow to update the virtual limit $x^a_{lim}$.

At the beginning of the event (step 29), $obj^a$ memorizes the committed capacity. This variable will be used later to compute the reliability score of the agent.

The state variable represents the different stages of a load shedding event. When $state = 0$, the agent is idle. Receiving an order puts the agent in $state = 1$ until the shedding starts and then goes to $state = 2$. While $state = 2$, the amount of flexibility the agent has engaged in the collective effort directly impacts the consumption in the user's home. This has not been shown in the pseudo-code for clarity.

At the end of the event (step 34), the agent updates its history $H^a$ and its coefficients $k_1$, $k_2$ and $k_3$ then switches to $state = 3$ where it waits for the shared variables to converge before computing its grade. Self-evaluation computations described in section 3.3.3 are not integrated in this pseudo-code.

Turnover score computation (step 35) can be done immediately, as opposed to the capacity score $C^a$ and the reliability score $F^a$ which depend on variables that are collectively computed and therefore need some time to converge. This is why the agent waits $t_{wait}$ time steps while in $state = 3$ before computing $C^a$ and $F^a$ (steps 42 and 43).

The agent's mark $g^a$ can then be computed (step 44).

**Algorithm 1** Pseudo-code for an agent $a$

**t = 0**

$x \leftarrow f, s \leftarrow x, w \leftarrow 0, state \leftarrow 0$

**t > 0**

Let $M = \{(s^*, w^*, g^*_{max}, c^*_{min}, c^*_{max})\}$ be all messages sent to $a$ during round $t-1$

$s \leftarrow x_t - x_{t-1} + \sum_M s^*$, $w \leftarrow \sum_M w^*$

$g_{max} \leftarrow \max_M(g^*_{max})$

$c_{min} \leftarrow \min_M(c^*_{min})$, $c_{max} \leftarrow \max_M(c^*_{max})$

Let $J \subseteq N \backslash \{a\}$ be a set of $j$ agents randomly chosen

$\alpha_a \leftarrow \frac{1}{j+1}$

send $(\alpha_a s, \alpha_a w, g_{max}, c_{min}, c_{max})$ to $M \cup \{a\}$

$agg \leftarrow \frac{s}{w}$

**if** $O = \{Q, t_d, t_f\}$ is received **then**

  send $O$ to $j$ agents

  $x_{lim} \leftarrow \frac{g}{g_{max}} f^a$

  $x \leftarrow x_{lim}$

  $state \leftarrow 1$

**end if**

**if** $\left|\frac{agg}{Q}\right| \geq H$ **then**

  **if** $t - cnt_1 \geq t_{wait}$ **then**

    $x \leftarrow x\left(1 + \frac{Q-agg}{agg}\right)$

    $cnt_1 \leftarrow t$

  **end if**

**else**

  $cnt_1 \leftarrow t$

**end if**

**if** $x \geq x_{lim}$ **then**

  $x_{lim} \leftarrow x_{lim} + (f^a - x_{lim}) * v$

**end if**

$x_{lim} \leftarrow \min(x_{lim}, f^a)$

$x^a \leftarrow \min(x^a, x_{lim})$

**if** $state = 1$ and $t \leq t_d$ **then**

  $obj^a \leftarrow x^a$

  $state \leftarrow 2$

**end if**

**if** $state = 2$ and $t > t_f$ **then**

  update History $H^a$

  compute Turnover $T^a$

  update coefficients $k_1$, $k_2$ and $k_3$

  $cnt_2 \leftarrow t$

  $state \leftarrow 3$

**end if**

**if** $state = 3$ **then**

  **if** $t - cnt_2 \geq t_{wait}$ **then**

    compute Capacity $C^a$

    compute Reliability $R_a$

    $g^a = k_1 R^a + k_2 C^a + k_3 T^a$

    $state \leftarrow 0$

  **end if**

  $g_{max} \leftarrow \max(g, g_{max})$

**end if**

# 4. Experiments

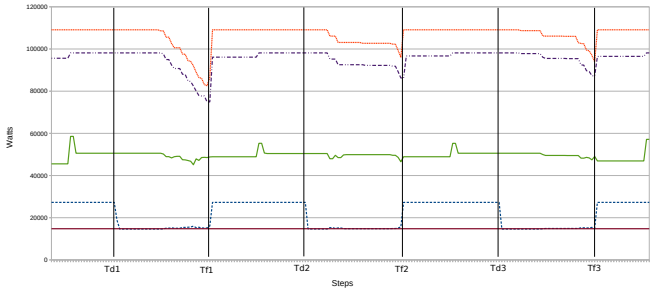
## 4.1. Test protocol

We evaluated our algorithm on realistic simulated data. In the following, we will describe this protocol.

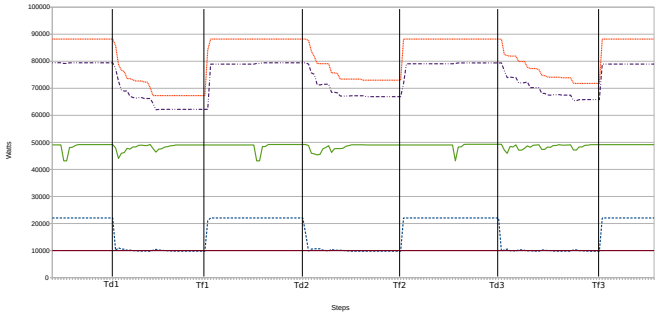
Each building is represented by an agent which possesses a fixed electricity consumption of 3500W at all time. It also has a flexibility (in Watts) randomly chosen in the set $\{100, 200, 400, 500, 2000, 3000\}$ (100W represents a few connected light bulbs and 3000W represents a water heater) , meaning an average flexibility of 1033W for the population, corresponding to a heater switching off with a slight dimming of the global lighting.

To simulate a realistic behaviour of the inhabitant, each agent is given a failure probability drawn from an exponential distribution. This is to reflect the fact that a lot of people will have a relatively good reliability, while some of them will systematically cancel their participation in the load shedding.

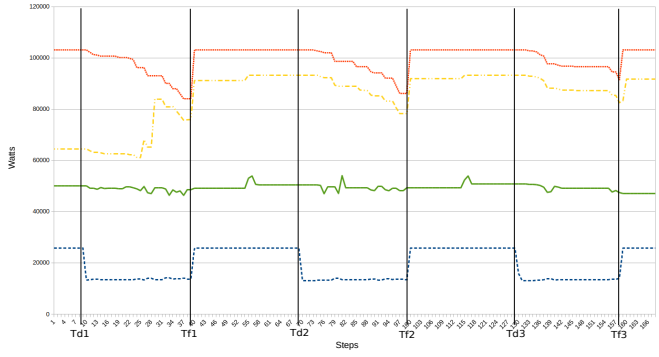
Before each event, the simulation determines if the agent will fail according to this probability. Once again, to simulate a realistic behaviour, we spread the agents failure over the time of the event, following a polynomial curve. To show different kinds of failure, we created two different behaviours. The first corresponds to users noticing a difference in temperature that are willing to turn their heaters up again, reducing the flexibility of their home. This results in a progressive failure of a part of the population, growing faster as time passes. In the simulation this is implemented by a polynomial repartition of the failure point of the agents. We later reference this scenario as the "heater scenario". The second behaviour is a lighting one in which users notice a dimming in their lightning, and turn them back up immediately. This is implemented by the inverted polynomial function we use in the "heater scenario", as the amount of agents failing will be high at the beginning of the event and quickly stabilise. The main difference between these two scenarii is the time when the user reacts to the shedding. In the case of the lightning, the users either notices it quickly or does not react at all, as in the heater behaviour a small temperature change cannot be noticed that fast thus leading to a slowly growing proportion of the agents reacting. We also ran tests with both scenario combined, applying the lighting behaviour to the agents with the lowest flexibility (as lower flexibility mostly comes from dimming lights only) and the heater behaviour to the agents with the highest one (as heaters offer a bigger shedding capacity). The initial weighting coefficient of the reliability score $k_1$ is set to $0.5$, the capacity one $k_2$ is at $0.4$ and the turnover coefficient $k_3$ at $0.1$. The agents start with a mark randomly chosen between 0 and 1 and their history is limited to 10 events. We set the hysteresis to $H = 0.02$, the delay $t_{wait}$ to one time step and the variation coefficient to $v = 0.5$. When not stated otherwise, each agent communicate with $j = 10$ other agents chosen uniformly at random at each step.




(a) Heater scenario

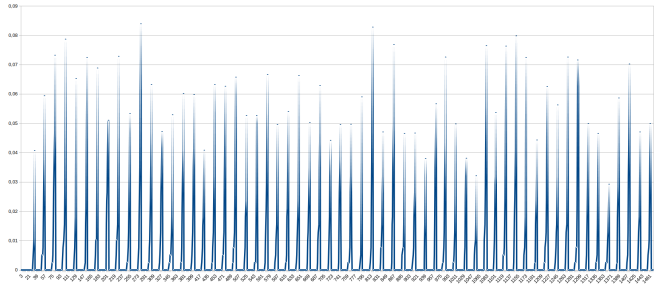



(b) Lightning scenario

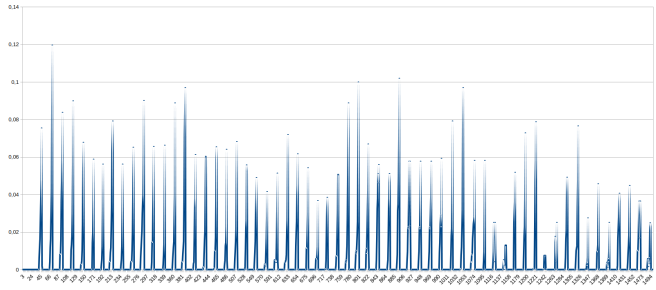



(c) Heater and lightning combined

Figure 3: These three charts show the performances of the system during three consecutive load shedding events. The red dotted line represents the total flexibility, most of the time followed by the sum of the $x_{lim}$ of the agents (in yellow dots and dash line). The continuous green curve is the aggregate. The dashed blue curve shows the total consumption, brought down to the same scale for more readability.



(a) Without self-evaluation



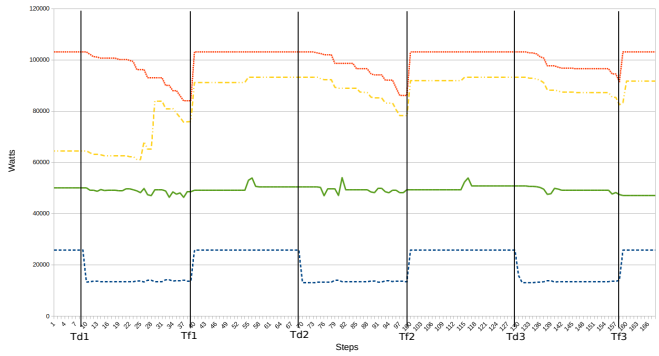
(b) With self-evaluation

Figure 4: This chart shows the failure ratio between the total decrease of flexibility at each step and the total committed energy that this decrease impacted. Each peaks is a load shedding event.

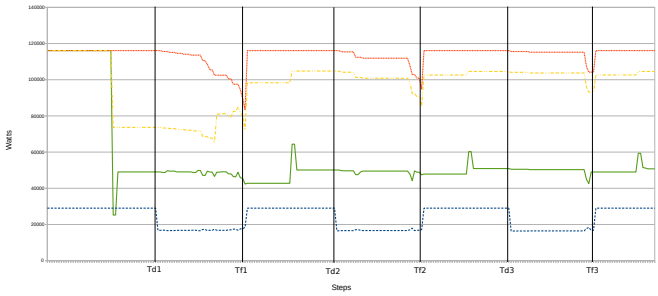
## 4.2. Results

Our simulations had two main objectives. First, to show the efficiency of a multi-agents system in maintaining the curtailment stability in a decentralized manner. Figure 3 shows the ability of our system to compensate for decreases in the agents flexibility in real-time. We see that as the total flexibility of the population plummet following the disengagement of a part of the population, the aggregated shedding capacity is maintained, suffering only from small variations.The different scenarii illustrate the robustness of the system with respect to the behaviour of the end users. Our system is able to maintain a stable curtailment and does not rely on the user's behaviour (the total flexibility has to be higher than the requested shedding capacity). Moreover, this quality is obtained without any supervising entity.

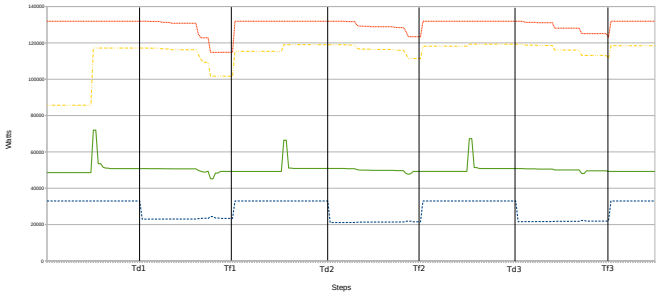
Then, to show the effect of the self-evaluation process on the curtailment quality, we compared the results obtained with and without the self-evaluation. we ran longer tests to allow the agents to gather enough information in their history to assign themselves a relevant mark. On figure 4(a), the agents self-evaluation is disabled. At each event, the agents commit themselves to their full flexibility. We see that the amount of engaged energy impacted by the changes in the flexibility of the agents stays the same. On figure 4(b), the self-evaluation is enabled. The ratio trend of impacted capacity over total flexibility loss diminishes over time instead of staying high, showing the ability of the self-evaluation system to optimise each agent's share .




(a) $j = 10$




(b) $j = 5$




(c) $j = 2$

Figure 5: These charts show the performance of the system with different connectivity constraints. The legend is the same as figure 3.

Finally, we verified the efficiency of the propagation algorithm by trying different connectivity levels on the agent to ensure that the gossip consensus process worked in various condition, even with limited connections between the agents. Figure 5 shows the performances of the system when each agent communicates with 10, 5 or 2 neighbours at each step. We see that the impact of the connectivity constraint is barely noticeable, as it really only impacts the convergence speed when the target value is far from the current value.

## 5. Conclusions and perspectives

Distributed load shedding is prone to quickly develop in the upcoming years. As a straightforward solution to the biggest current problems of the energy sector without major drawbacks, it offers an ecological way to keep the network running while allowing consumers to save energy and money. The biggest obstacle faced by existing systems and ongoing work is to be able to deploy such a system while not affecting the user's comfort with a system simple enough to be reliable on a large scale. Yet, in a centralized system, taking the user's comfort into account implies a constant and heavy two-way communication between the server and the nodes, limiting the system's scalability and flexibility.

In this paper, we presented an original decentralized architecture to solve this problem. Following a bottom-up approach, the user is the starting point of the process and stays in control at all time. The home energy management system deduces the energy the building can shed and transmit it to the system we built on top of it. By doing so, we guarantee the user's comfort at all time, as opposed to more demanding system like price-based ones usually presented in the literature. We then use a gossip based communication protocol between the agents to transmit the information needed for the process to work. It provides a robust and scalable decentralized architecture allowing the agents to quickly reach a consensus on their respective participation to the effort. During an event, the simple reactive behaviour of the agents ensures a stable curtailment, as any failure of an agent is immediately absorbed by the adjustment of many others. This adaptability is then improved by a self-evaluation process which ensure the fair sharing of the effort between the agent, depending on their reliability, capacity, and frequency of participation. Event after event, the population regulates itself to improve the quality of the curtailment. This self-evaluation process is then fine-tuned by a local optimisation process that adjusts the weighting coefficients on the go, considering the latest performance of the whole system (always in a decentralized manner).

Moreover we let the on-site system take care of the user and independently determine its share on a load shedding effort, separating the complexity and efficiency of the process from the number of partakers. It allows a fully incremental deployment of the system, whereas a simple stochastic approach cannot work without a minimal number of connected users. In our system, a single user can curtail its load if needed, as any declared flexibility can immediately be used by the system.

Our simulations on realistic data confirm the resilience and the self-adaptation ability of our system to guarantee a stable curtailment even when any agents are disengaging at once. They also show the efficiency and relevance of the self-evaluation mechanism to improve the quality of the shedding over time, underlying the need of preliminary benchmarking to provide a high quality of service from the beginning.

These promising results pave the way to further works on this model.

First, we have to test our system in real-life settings with more demanding situations, in terms of failure connectivity

constraints, failure probability and scalability. It will help us determine the limitations of the model depending on the use cases.

Second, we plan to add a local parameters optimisation process similar to the one used for $k_1$, $k_2$ and $k_3$ that could be used by the agent to adjust the value of the hysteresis $H$ or the timer $t_{wait}$ depending on their evaluation of the system efficiency. Too many crossing detection could slow down the system as well as irrelevant detections caused by a too small $t_{wait}$.

Third, it would be interesting to change the way the load shedding order is provided to the population. It is currently sent to one or more agents by a unique external source which is considered reliable. The entire system is therefore tied to this source (the energy provider in our case). We think that a learning mechanism coupled to a trust network could allows the agents to predict curtailment needs, to anticipate an order or to react autonomously without binding to an external operator. This would greatly enhance the system resilience.

## References


[1] Jacob Beal, Jeffrey Berliner, and Kevin Hunter. Fast Precise Distributed Control for Energy Demand Management. In *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems*, pages 187–192. IEEE, September 2012.

[2] D S Callaway and I A Hiskens. Achieving Controllability of Electric Loads. *Proceedings of the IEEE*, 99(1):184–199, January 2011.

[3] Specification Technique Edf. *Materiels emission et de reception de telecommande centralisee a frequence musicale*.

[4] Sylvain Frey, Ada Diaconescu, David Menga, and Isabelle Demeure. A holonic control architecture for a heterogeneous multi-objective Smart Micro-Grid. *International Conference on Self-Adaptive and Self-Organizing Systems, SASO*, pages 21–30, 2013.

[5] Ju Hang Ju Hang, Jin-Xin Tian Jin-Xin Tian, and Huang-Gui Lin Huang-Gui Lin. Application of Artificial Neural Network in Intelligent Building. In *2007 International Conference on Machine Learning and Cybernetics*, volume 7, pages 4215–4220. IEEE, 2007.

[6] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Proceedings. 44th Annual IEEE Symposium on Foundations of Computer Science, 2003.*, pages 482–491. IEEE, 2003.

[7] Amir-Hamed Mohsenian-Rad, Vincent W. S. Wong, Juri Jatskevich, Robert Schober, and Alberto Leon-Garcia. Autonomous Demand-Side Management Based on Game-Theoretic Energy Consumption Scheduling for the Future Smart Grid. *IEEE Transactions on Smart Grid*, 1(3):320–331, December 2010.

[8] Alex Papalexopoulos, Jacob Beal, and Steven Florek. Precise Mass-Market Energy Demand Management Through Stochastic Distributed Computing. *IEEE Transactions on Smart Grid*, 4(4):2017–2027, December 2013.

[9] Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nick Jennings. Agent-based control for decentralised demand side management in the smart grid. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[10] Vinayak V Ranade. *Model and control for cooperative energy management*. Massachusetts Institute of Technology, 2010.

[11] Vinayak V. Ranade and Jacob Beal. Distributed Control for Small Customer Energy Demand Management. In *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 11–20. IEEE, September 2010.

[12] Mohammad R. Vedady Moghadam, Richard T B Ma, and Rui Zhang. Distributed Frequency Control in Smart Grids via Randomized Demand Response. *IEEE Transactions on Smart Grid*, 5(6):2798–2809, November 2014.

[13] Fetahi Wuhib, Mads Dam, and Rolf Stadler. A gossiping protocol for detecting global threshold crossings. *Network and Service Management, IEEE Transactions on*, 7(1):42–57, 2010.

[14] Haibo You, Vijay Vittal, Juhwan Jung, Chen-Ching Liu, Massoud Amin, and Rambabu Adapa. An intelligent adaptive load shedding scheme. *Proc. 2002 14-th PSCC*, pages 17–6, 2002.