Context Monitoring Optimization for Demand Response

Keywords: Demand Response, User Profiling, Context Monitoring

Demand Response (DR) describes the reduction of energy consumption by end-use customers in response to power grid needs, economic signals or special rates. According to the Federal Energy Regulatory Commission, DR is defined as: “Changes in electric usage by end-use customers from their normal consumption patterns in response to changes in the price of electricity over time, or to incentive payments designed to induce lower electricity use at times of high wholesale market prices or when system reliability is jeopardized.” - [1]

Developments in DR usually reflect national conditions and are triggered by different policies however they are all based on the same paradigms: Indirect Load Control, where the consumer reacts to external stimuli and adjusts his consumption (e.g. new price) or Direct Load Control, where specific devices are automatically adjusted according to specific agreement between the consumer and the provider. Although operational in theory, in practice, both approaches suffer from significant drawbacks; direct control disregards user behavior and habits while indirect approaches necessitate considerable interaction with the customer, thus creating discomfort.

It is evident that a new approach is required that will combine the salient features of both direct and indirect DR paradigms so as to appropriately adjust energy consumption while in parallel minimize discomfort and intrusion. This last requirement implies the design and implementation of automated tools that exploit explicit and/or implicit information and limit direct interaction with the users. The latter necessitates the dynamic definition of a user profile based on the aggregation of various information sources (e.g. questionnaires, building sensors, and actuators), the analysis of user-driven events as well as the reaction of the user to external -system generated- stimuli.

The envisaged paradigm shift calls for a personalized framework that operates in the consumer’s residence. The profiling engine should be deployed in a low end controller device (e.g. Raspberry Pi - [3], Soekris - [2] or similar embedded PC), not larger than an ADSL router and constantly monitor the residence of the consumer.

Data acquisition performed by the monitoring process is an essential part of this profiling mechanism. The rate of sampling is a crucial factor since it is related to:

i) the successful/unsuccessful detection of events,
ii) the processing power needed to perform the sampling and,
iii) the energy that the device and the sensor nodes consume during such actions.

In order to address these issues we designed a simple and efficient mechanism that dynamically adapts the sampling rate of the monitoring procedure. The algorithm attempts to kill two birds with one stone; on one hand, in the absence of events minimize the number of unnecessary monitoring actions, while on the other hand detect –ideally all– events when they appear.

From a high level, methodological point of view, upon initiation, the algorithm searches for events scanning at the highest possible frequency (e.g. one scan per designated time). In the absence of events, its scanning rate is progressively reduced until a predefined, low threshold is reached. When an event is identified the algorithm is re-calibrated to the highest possible scanning frequency and keeps measuring at this frequency until no event is monitored. The same process is repeated until all events have been identified.

The merits of the mechanism have been quantified by means of an analytical model. We model the procedure as a Markov chain, each state of which essentially adjusts the scanning frequency of the monitoring procedure. We assumed that a scanning/sensing/monitoring attempt is successful if it identifies an event, and unsuccessful otherwise. Each state of the Markov chain is characterized by a monitoring frequency. If the algorithm detects the occurrence of an event, it transitions to state S_k in which it operates at the highest possible monitoring frequency. If it does not detect an event, it transitions to a 'lower' state (towards state S_1), in which sensing is sparser. Transition probabilities for any state S_i are P_i and 1- P_i where the former denotes the probability of transition from S_i to S_k and the latter from S_i to S_{i-1}.

Further analysis indicates that the key factor that should be taken into account is the density of events d, i.e. the frequency of events occurrence during a day. After identifying d, then we can precisely calculate the length of the chain and thus calibrate the algorithm so as to detect the highest amount of events with the minimum possible number of loops.

The added value of the aforedescribed algorithmic solution is twofold. First and foremost, it achieves a significant reduction in the CPU load of the monitoring device. CPU-load reduction is extremely important in our case since the embedded devices that accommodate the profiling framework are equipped with a single CPU (i.e. a Raspberry Pi comes with a 700MHz ARM processor) thus performance optimization is a strict pre-requisite. It is worth mentioning that theoretic analysis indicates that in eventful conditions, we can monitor more than 70% of events by performing 50% less loops than the always-on-case. Additionally, the generic nature of the mechanism enables its application on any case that can be reduced to an event-detection use case. From an end-user perspective the benefits are indirect and appear due to the minimization of the User-System interactions.

Our work now focuses on developing a learning scheme so as to enable the algorithm to take into consideration past information and seasonality in order to autonomously calibrate its input parameters without requiring any human intervention. Obviously, each human is unique, thus a one-size-fit-all solution is not adequate. Our plan is to design an algorithm that will be adaptable to the end user, learn from him and eventually, after a period of time, identify and adapt to its behavioral (in terms of events generation) pattern.

[1] Balijepalli, Murthy; Pradhan, Khaparde (2011). "Review of Demand Response under Smart Grid Paradigm". IEEE PES Innovative Smart Grid Technologies.

[2] Soekris Engineering: http://soekris.com/

[3] Raspberry Pi: http://www.raspberrypi.org/