A DEMAND RESPONSE SYSTEM FOR HIERARCHICALLY ORGANIZED AGGREGATORS IN SMART GRIDS

Sergios Soursos[(a)], Vassilis Kapsalis[(b)], George Petropoulos[(a)], Yiannis Karras[(c)] and Loukas Hadellis[(b)]

[(a)] Intracom S.A. Telecom Solutions, R&D Unit, Telco Software Business Division, 19002 Peania, Athens, Greece
[(b)] Technological Educational Institute of Patras, 26334 Patras, Greece
[(c)] inAccess S.A., 15125 Maroussi, Athens, Greece

[(a)]{souse, geopet}@intracom.com, [(b)]{kapsalis, loukas}@teipat.gr, [(c)]jkarras@inaccess.com

ABSTRACT

Current advances in Smart Grids have reshaped the business ecosystem of the energy market, allowing for the role of an aggregator to emerge. At the same time, the need to deal with power shortages and blackouts has rendered the participation of consumers to the management of energy quite necessary. Demand Response practices are becoming more popular and relative standards like the Open Automated Demand Response (OpenADR) have emerged as very promising technologies for the Smart Grid. In this context, we present an architecture that introduces the demand response functionality in an environment of multi-level hierarchically organized aggregators. We adopt and extend the OpenADR standard and provide the required functionality to support such a system. We describe the comprising components and interfaces and present the technologies used for the implementation of the system.

Keywords: demand response, automation, aggregator, multi-level hierarchy

1. INTRODUCTION

The Smart Grid concept and its supporting technologies were introduced in an attempt for society to decrease the consumption of energy resources, especially during periods of critical consumption levels and/or reduced energy production. The need for detailed monitoring of the consumption and for prediction of the future demand have led to the update of the infrastructure of the Distribution System with smart meters, comprising the Advanced Metering Infrastructure (AMI). With the placement of smart meters, the Independent System Operator (ISO) will be able to foresee demand peaks and take the appropriate measures so as to avoid overloading the Distribution System.

With the introduction of Smart Grids, Demand Side Management (DSM) has emerged as a new field of energy management that aims at controlling/shaping the demand for energy, while considering the status of the transmission and distribution networks and the available energy resources. Demand Response (DR) is a DSM approach that manages electricity demand by sending economic or incentive-based signals to the consumers who react by altering their consumption behavior based on terms of the DR program(s) they have subscribed to.

In this context, a new business role is emerging in the Smart Grids ecosystem; that of the aggregator, a business entity that acts as a mediator/broker between consumers and the Utility Operator or the ISO (Gkatzikis, Koutsopoulos and Salonidis 2013). Representing a large number of consumers provides the aggregator the bargaining power to negotiate with the ISO on prices. On the other hand, the aggregator receives DR signals from the Utility/ISO, which has to process and re-distribute them down to its customers so as to achieve the requested power cuts, considering the customers' constraints and comfort levels as well as trying to attain monetary gains for itself and its customers by minimizing also the aggregation risks that may result in penalties.

In this paper, we envision the presence of more than one aggregators per district, organized in a multi-level hierarchical structure that compete (same level) and interact (adjunct levels) with each other, so as to achieve optimal and dynamic DSM, with monetary gains both for the consumers as well as for the aggregators. It becomes obvious that an aggregator with such a rich portfolio of offered services requires an advanced system that allows for all the interactions to take place and has built-in intelligence to support the automatic handling of DR events, programs, subscriptions and constraints. Hence, in this environment, we propose an ICT system, named DAMAZO, which enables the materialization of the aforementioned concept, enabling the communication between aggregators, as well as allowing for different DR programs to be supported and different DR policies to be implemented.

The remainder of this paper is organized as follows: Section 2 overviews the existing works in the area of DR automation; Section 3 provides the core concept of the DAMAZO system, the programs and the mechanisms that it currently supports; Section 4 describes the architecture of the DAMAZO system; Section 5 provides an insight on the algorithms designed and deployed while Section 6 provides the concluding remarks and the future work.

2. RELATED WORK

The energy crisis of 2002 in California served as the driving force for the Demand Response Research Center operated by Lawrence Berkeley National Laboratory to create the first version of the Open Automated Demand Response (OpenADR v1.0) specification, which was released in April 2009 (Piette, Ghatikar, Kiliccote, Koch, Hennage, Palensky, and McParland 2009). This specification describes an open standards-based communications data model designed to promote common information exchange between the Utility/ISO and electric customers using demand response price and reliability signals. The intention of the data model is to interact with building and industrial control systems that are pre-programmed to take action based on a DR signal, enabling a demand response event to be fully automated, with no manual intervention. The DR system comprises of a single server denoted as Demand Response Automation Server (DRAS), which communicates with the corresponding DRAS clients, enabling a demand response event to be fully automated. Although, the specification promoted interoperability between Utility/ISO and electric customers, it nevertheless lacked a multi-level hierarchy between a DRAS and its corresponding clients.

The OpenADR Alliance, a mutual benefit corporation which was created to foster the development, adoption, and compliance of the OpenADR Smart Grid standard, has recently released the OpenADR 2.0a and 2.0b (draft) profile specification and schema (OpenADR 2.0a, 2012) (OpenADR 2.0b, 2013), which consist part of OASIS Energy Interoperation Specification 1.0 (OASIS EI 1.0, 2012). A feature of the new version is the adoption of a hierarchical structure between Virtual Terminal Nodes (VTNs) and Virtual End Nodes (VENs) (see Fig. 1), according to the concepts introduced in a white paper by EPRI (EPRI, 2010).

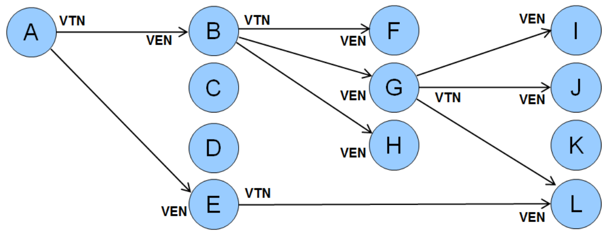

Figure 1: Example of DR interactions under OpenADR v2.0

In the above figure, certain Parties (B, E, and G) act as both VTN and VEN. This directed graph with arrows from VTN to its VENs could model a Reliability DR Event initiated by the Independent System Operator A who would invoke an operation on its second level VTNs B-E, which could be a group of aggregators. The second level VTN B, in turn invokes the same service on its VENs FGH, who may represent their customers or contracted resources. However, OpenADR 2.0 does not define how the nodes react to the information. In nodes, which support both the VTN and VEN interfaces (e.g., aggregators) there are no specifications or constraints on how messages arriving at the VEN interface is coupled or translated into any subsequent messages that may be sent from the VTN interface and vice versa.

A project dealing with the Demand Response concept is ADDRESS, a 5-year large-scale R&D project launched in 2008, that aims to deliver a comprehensive commercial and technical framework for the development of “Active Demand” (AD) in the Smart Grids of the future (Valtorta and Giovanni, 2011). “Active Demand” is the term used instead of “Demand Response” to describe the participation of consumers in the management of energy resources. The project also identifies the role of the aggregator as a key role in the energy market ecosystem. Although the work of ADDRESS is of wider scope, several objectives of ADDRESS are partly in line with our work. However, not much information about the project’s proposed architecture and its implementation is provided, thus not much can be said about the compatibility of our system with the approach of ADDRESS. Furthermore, our objective for a multi-level hierarchical structure of aggregators is, as far as we know, not supported by ADDRESS.

Beywatch was a project funded by the European Commission under FP7, that aimed to design, develop and evaluate an innovative, energy-aware, flexible and user-centric solution, able to provide interactive energy monitoring for white goods, intelligent control and power demand balancing at home, block and neighbor level (Beywatch D2.1 Service Requirement specification, 2009). The BeyWatch concept included a hierarchical network architecture of interactive metering and intelligent control devices: a) the Agent at Home / Office level, b) the Supervisor at building / square / neighborhood level and c) the Service Centre at the utility level. The proposed system introduced a two layers hierarchy: micro-management and medium-management level. Under the micro-management level, all the devices in the home or a building were set under local interactive monitoring and intelligent control, in order to achieve amortization of loads and peak suppression of small-scale power consumption. The local control elements were included in a hierarchical system that coverer larger geographical regions (e.g. building blocks or neighborhood) that enabled medium-level control and coordination of the energy resources.

The SmartHouse/SmartGrid is another EU funded FP7 research project, which exploited the potential that is created when homes, offices and commercial buildings are treated as intelligently networked collaborations. The project envisioned a system, where SmartHouses are able to communicate, interact and negotiate with both customers and energy devices in the local grid, resulting into a more efficient operation of the electricity system, because consumption can be better adapted to the available energy supply, even when the proportion of variable renewable generation is high (Warmer et al., 2009). A commercial aggregator could exercise the task of jointly coordinating the energy use of the SmartHouses or commercial consumers that have a contract with him.

3. CONCEPT AND SYSTEM OFFERINGS

As already mentioned, our system follows the OpenADR v1.0 specification. In the OpenADR, the core entity expected to materialize the DR policies is the DRAS. DRAS is expected to be deployed in a three-level hierarchical topology: on the top level we have the existing system of the ISO or Utility Operator, which from now on we refer to as “Back Office”; in the middle level we have the DRAS (operated either by the Utility/ISO or by an aggregator) and in the bottom level we have the consumers.

A brief overview of the standard DR procedure is described here: initially, the interested consumers are subscribed to the DR programs they find appealing. This subscription is made at the DRAS, but the Back Office is informed as well. At some given point in time where a need for activation of a program is required (e.g. due to a critical situation at the distribution network), the Back Office issues a respective DR event to the DRAS. The DRAS, in turn, finds the consumers subscribed to the respective program and forwards the event. Consumers are informed about the event and either manually take the necessary actions (as dictated by the program) or a software agent takes the responsibility of fulfilling the expected actions.

In our work we introduce the *eDRAS (enhanced DRAS)* that implements the core functionality as specified by the standard, as well as some extensions that allow it to be used in more complex topologies and to offer more enriched functionality. Below, we highlight the offerings of the eDRAS. The innovation of our work stems from the realization of an end-to-end multi-hierarchy DR system (DAMAZO) aligned with the architecture proposed by OpenADR 2.0 (regarding to the multi-layered client-server VTN-VEV structure), introducing business and logic rules implemented in each layer for the end-to-end handling of the DR events.

3.1. Multi-level hierarchical architecture

The standard DRAS offers three types of interactions: i) with Back Office; ii) with the Smart Client (DR-aware client) and iii) with the Thin Client (non DR-aware client). For the eDRAS to support the multi-level hierarchical architecture, as shown in Figure 2, a new interface is required; that of the inter-eDRAS communication.

To support this new type of interaction between adjacent eDRASes, we re-used the interfaces specified by the standard between the DRAS and the Smart Client. Actually, for an eDRAS of level *n*, the eDRAS located at level *n+1* can be considered as a Smart Client as well. In this simple way, a multi-level topology is supported and only modifications in the logic residing inside an eDRAS are required.

3.2. Types of DR programs

Traditionally, DR supports two categories of programs which the consumers interested in participating can subscribe to: the price-based DR programs and the incentive-based DR programs. The eDRAS supports the following programs:

- Base Interruptible Programme (BIP): the participant is asked to decrease its consumption and, in case he acts accordingly, receives a compensation (in terms of discounts or credit) that is specified in the contract.
- Time-of-Use + Critical Peak Pricing (ToU+CPP): the participant is provided with a different unit price depending on the time of day, while during critical situations he receives a relative update of the price (higher price typically leads to decreased demand).
- Real Time Pricing (RTP): the participant receives updates on the price that are valid till the next update is received. Again, the consumer is expected to react to increased prices.
- Direct Load Control (DLC): the participant grants to the Utility/ISO or to the aggregator the ability to remotely control his appliances. Once such an event is received, the respective appliances are switched off/on, or their operation is shifted at a later time. Refusal to adhere to the event is penalized.

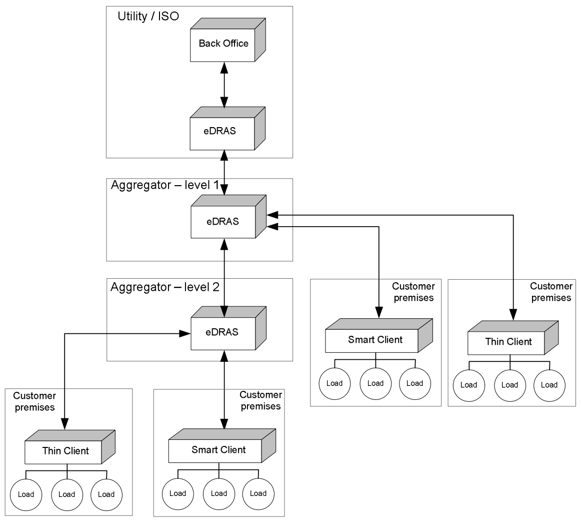

Figure 2: eDRAS’ support for multi-level hierarchical architecture

The existence of multiple eDRASes introduces a complexity in handling the different programs, since an eDRAS is at the same time a client for the upper level eDRAS. And one cannot expect that each eDRAS operator (i.e., aggregator) offers exact the same programs to its clients. Assume that an *n* level eDRAS is subscribed (as a client) to a BIP program that requests a decrease of 10 kW between 10 am and 11 am. At the same time this eDRAS has issued a BIP program to its *n+1* level clients that requests a decrease of 2 kW between 10 am and 11 am. In case of a BIP event from the *n-1* level eDRAS, the *n* level eDRAS has to map the upper level BIP program to its own BIP program (straightforward in this case: issue at least 5 BIP events to its *n+1* level clients in order to meet the requirement from the upper level).

What happens however when the *n* level eDRAS has subscribed to a BIP program but has not issued any BIP program for its clients? How the eDRAS should react upon the reception of a BIP event in this case? It becomes obvious that the mapping between different types of programs is not straightforward. To deal with this situation we have come up with the following relationships between types of programs.

Table 1: Relationships between DR programs

Received Event's Program	Compatible Program	Complementary Program
BIP	BIP, DLC	ToU+CPP
ToU+CPP	ToU+CPP, RTP	BIP
RTP	RTP, ToU+CPP	BIP
DLC	DLC	BIP

When an event is received, the eDRAS tries to resolve the request by issuing an event of the same program type (differences in the parameters with respect to amount and time are resolved). If not such a program exists, clients subscribed to compatible programs are addressed. In such a case the risk of not fulfilling the needs of the original event are expected to be low and depends on the number of clients addressed and the overlaps in time periods. In the improbable case where no compatible programs exist, then events belonging to complementary programs can be issued, but the risk of not fulfilling the original requirements increases. It is expected however, that a rational eDRAS operator will offer to his clients programs that are of the same type with the ones that he has already subscribed to, or the other way around.

3.3. Selection of clients, monitoring and statistics

A client that subscribes to a program can also submit his time constraints. For example, a client subscribed to a BIP program that runs from 10 am to 12 am, may have a one-hour constraint. This can be known a priori and expressed during the subscription or can be declared dynamically (in this case the client *opts-out* from the specific program for a given time). Such time constraints and others related to the location of the clients, the groups they belong to, etc., must be considered by the eDRAS when issuing an event.

Moreover, the eDRAS should be able to estimate, before issuing an event, whether a client can accept it, considering the active events he has already accepted, the current and average consumption levels, available shed prediction throughout the day, etc. Further criteria for selection include performance and fairness. Performance has to do with the reaction of a single client after the reception of an event; if he had managed to save some energy and how close to the target value he performed. Fairness has to do with the fact that not the same clients should be selected all times; even though they might be top ranked considering their performance. Such statistics need to be collected by the eDRAS through the appropriate monitoring mechanisms.

From the above, it becomes obvious that the selection of the appropriate clients to send an event is a complex procedure that requires lots of information so as to render the action taken successful or, at least, of low risk for failure.

4. ARCHITECTURE

Having outlined the core functionality of the eDRAS, in this section the architecture of the DAMAZO system is presented. The implementation of the eDRAS and the required interfaces has been based on the OpenADR v1.0 specification with proper extensions wherever required.

Figure 3 depicts the DAMAZO system architecture. The main entities presented are the eDRAS, the Smart Client and the Thin Client. The distinction between the Smart and the Thin Client is that the former is able to communicate with the eDRAS using the OpenADR-based messages (DR I/F) and has some intelligence of its own (Control Logic). It is usually collocated with a smart meter and is in the form of a gateway with programmable capabilities. The Thin Client on the other hand is a simple gateway that can receive commands in a specific format and has no additional processing capabilities other than operating as a simple load interface.

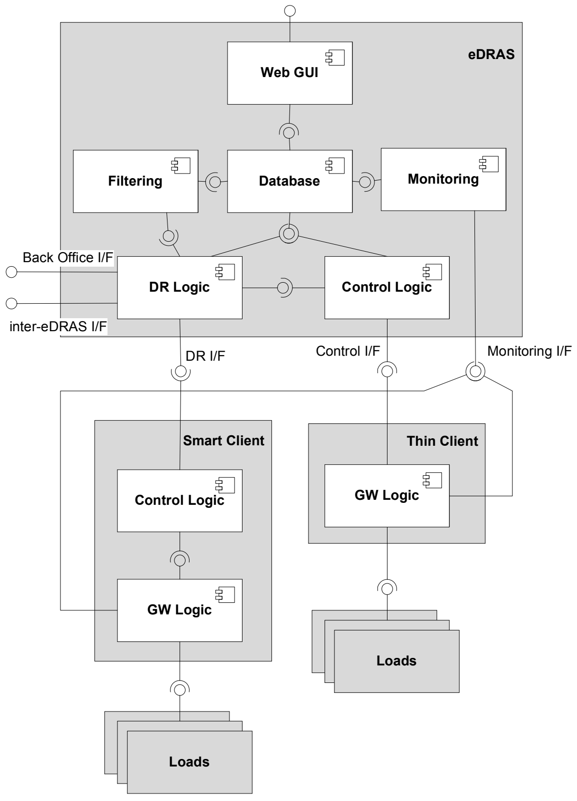

Figure 3: The DAMAZO System Architecture

The eDRAS is implemented as a Java EE application deployed in a JBoss Application Server. It consists of the following components, which are mainly implemented as Enterprise Java Beans (EJBs):

- Web GUI: Allows the eDRAS operator to create programs and register clients through the use of typical HTML pages. For each client further information for available loads and user preferences may be recorded, based on the level that the eDRAS resides. All respective information is stored at the database.
- Database: Stores all the required information received by the Web GUI component, as well as from the Monitoring, Control Logic and DR Logic components. It is implemented using the MySQL RDBMS.
- Monitoring: Collects data from the clients regarding the shed available, shed attempted (after the reception of a DR event), the current usage and the future usage. It provides a REST interface for Smart and Thin Clients to send their monitored data and saves them to the database, after performing some aggregation and normalization.
- Filtering: Implements the selection of candidate clients, given a specific DR event. It retrieves the client constraints from the database and compares them to the event requirements. Finally it provides the resulting list to the DR Logic component.
- DR Logic: Is the core component of the eDRAS. It receives a DR event from the Back Office or the upper level eDRAS, through the respective interfaces implemented using SOAP protocol. It queries the Filtering component for the list of candidate clients and then ranks them considering the criteria of performance and fairness. Then it splits the received event to the appropriate lower level DR events, running an allocation algorithm. Depending on the type of the selected clients, it either sends the resulting events to lower level eDRASes (through the inter-eDRAS interface), to Smart Clients (through the respective DR interface) or to the Control Logic interface (in the case of thin clients).
- Control Logic: it handles the details of the registered loads. Given a DR event, it decides which specific loads to either switch off or shift in time so as to achieve the requested target. Through the Control interface (implement with REST) it sends the load commands to the Thin Client.

The Smart Client is implemented as a stand-alone infrastructure on site, mainly due to the vast variety of equipment that may exist in the installation, storage needs to avoid data loss, scarce bandwidth and finally autonomous operation of field applications and logic. This infrastructure, consisting mainly of an embedded controller, undertakes the responsibilities of:

- Interfacing with all control and monitoring equipment present in the installation, in order to acquire data irrespective of the connection method and/or protocol.
- Reducing the volume of data required for transmission to eDRAS to something representative, yet less bandwidth demanding.
- Conveying events that are generated directly by the equipment upon their occurrence and generating events that are necessary yet not implemented within the equipment in the field.
- Storing all necessary parameters and events in a cyclical manner in the case of failure of the communication with the outside world.
- Implementing all necessary control loops that have to operate locally in an independent manner, providing a local autonomous implementation of intelligent DR algorithms.
- Interfacing with eDRAS using the respective standardized DR protocols.

The control and monitoring equipment as well as the loads connected are registered in the Smart Client during an initial provisioning procedure with minimal database functionality required in the field. The various components of the Smart Client have been implemented using C, C++ and Python over an embedded version of the GNU/Linux adapted for the RSC controller family. With regard to local control protocols used for communicating with the equipment, these vary depending on the technological capabilities of each component. Implemented interfaces include IEC 62056-21 over serial for local metering, Zigbee/IEC 802.15.4 for sub-metering and smart plugs control, as well as Modbus over RS-485 technologies as a common industrial and building infrastructure technology.

Finally, the Thin Client is implemented as a standalone embedded controller of lower cost that provides a subset of the functionality of the Smart client, mainly without providing local control loops and intelligent algorithms implementation or standardised communication to eDRAS. The Thin Client corresponds to installations with existing automation controllers as the only means for eDRAS to interface with the DR site, providing proprietary communication protocols and interfaces as well as custom information modelling. Actually, a Thin Client operates as a load interface, without any intelligence, other than acting as an interface between the eDRAS and the controlled loads. It is implemented as a RESTful service, running on .NET and implemented using C#. The communication between the eDRAS and any Thin Client may be based on proprietary protocols (e.g., GSM/GPRS, OPC, etc.).

5. DR AND CONTROL LOGIC

Much of the innovation introduced is included in the DR Logic and Control Logic components, as presented in the DAMAZO system architecture. They encompass all the required functionality and algorithms to achieve the promised offerings. In this section we will focus on certain mechanisms that are employed by the system.

5.1. Constraint-based Filtering

As mentioned earlier, the OpenADR standard foresees a number of stakeholders, namely the aggregator, the participant and the client. Moreover, it offers a variety of programs. Participants, clients and programs come with a set of properties accompanied by constraints mostly related to time duration and occurrence limitations. Appropriate actions are expected when a DR event of a specific program arrives at the DRAS for specific participants and clients.

DR logic is responsible to i) reject those programs, participants and clients that do not match the event properties and ii) decide and return the candidate clients and the available shed levels they can offer. The entire processing of filtering is summarized in Figure 4.

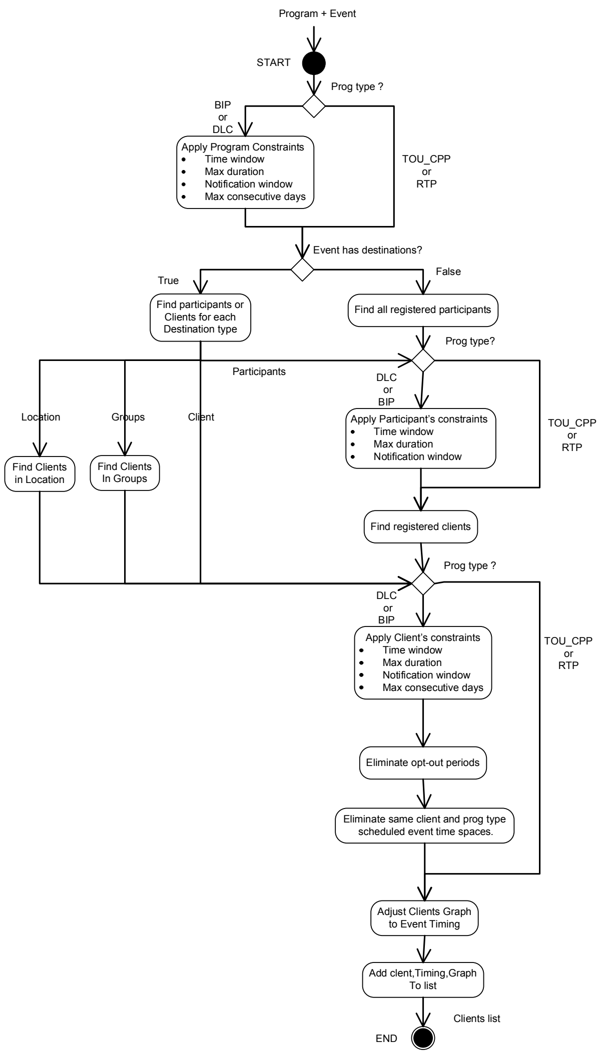

Figure 4: The filtering algorithm

The OpenADR standard specifies four types of constraints: ACCEPT (accept the event as it is), REJECT (reject the event if it does not comply with the constraint), FORCE (impose the constraint restrictions) and RESTRICT (find the intersection between the event and the constraint). Since one event can conflict with more than one constraint, we have come with the following rules to resolve the conflict between two constraints, with the goal to maximize the number of matched constraints. Note that we have also provided guidelines in the case of multiple constraints, but they are not included here due to space limitations.

Table 2: Resolution of an event's conflict with two constraints.

Constraint 1	Constraint 2	Result
ACCEPT	ACCEPT	ACCEPT
REJECT	REJECT	REJECT
ACCEPT	REJECT	REJECT
FORCE	FORCE	Choose the one with the longest duration.
RESTRICT	RESTRICT	Choose the one that provides the longest intersection.
FORCE	RESTRICT	FORCE
FORCE	REJECT	FORCE
ACCEPT	FORCE	FORCE
ACCEPT	RESTRICT	RESTRICT
RESTRICT	REJECT	RESTRICT

5.2. Ranking and Selection of Clients

DR Logic also implements the ranking and selection of the clients to be notified about the event as well as the allocation of sheds to each client, in case the event is not addressed to all clients and/or the shed offerings of the clients are more than the target of the event. As already mentioned, the ranking of the clients is based on fairness and performance criteria. Fairness is calculated according to the formula in (1):

$$F_i = 1 - \frac{TotalSuccessful\,\mathrm{Re}\,quests_i}{\max(TotalSuccessful\,\mathrm{Re}\,quests_k)} \qquad (1)$$

where *TotalSuccesfulRequests* is the number of positive answers of client *i* to the allocated shed (either through feedback or opt-outs). The performance criterion is calculated as follows:

$$B_i = PositiveOp\,ts_i \cdot Avg\,(Efficiency\,)_i \qquad (2)$$

where PositiveOpts is the percentage of positive answers from the client and Avg(Efficiency) is the average shed performed by the client in previous allocations.

$$Efficiency = \frac{CurrentShed}{ExpectedShed} \qquad (3)$$

Hence, the ranking for client *i* is calculated as:

$$Rank_i = w \cdot F_i + (1-w) \cdot B_i \tag{4}$$

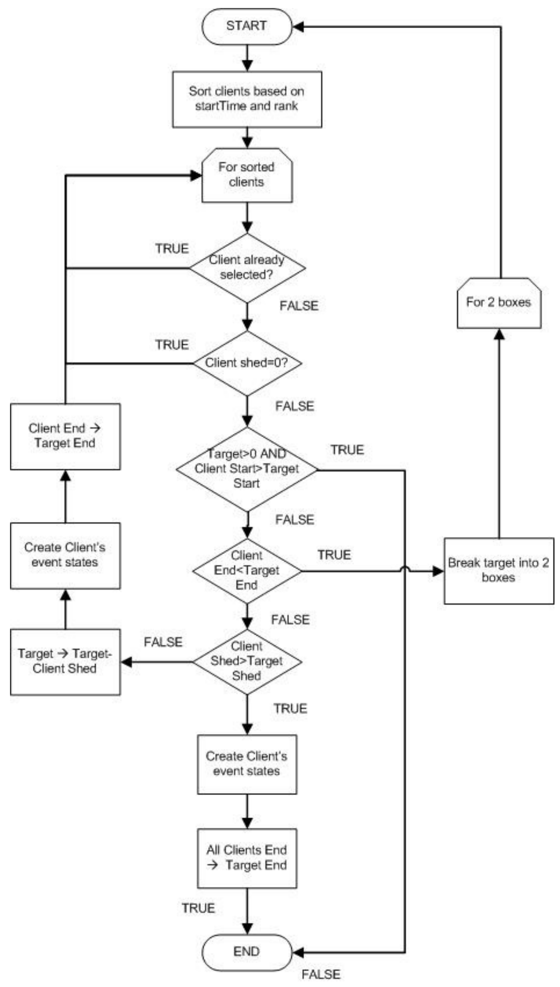
START
Sort clients based on startTime and rank
For sorted clients
Client already selected?
TRUE
FALSE
Client shed=0?
TRUE
FALSE
Target>0 AND Client Start>Target Start
TRUE
FALSE
Client End<Target End
TRUE
FALSE
Client Shed>Target Shed
FALSE
TRUE
Target → Target-Client Shed
Create Client's event states
Client End → Target End
Create Client's event states
All Clients End → Target End
TRUE
END
FALSE
Break target into 2 boxes
For 2 boxes

Figure 5: The selection and allocation algorithm

Once the clients are ranked, the selection and allocation algorithm comes in play. The selection considers the ranking as well as the time period in which the clients can offer the stated shed (as a result of the filtering process). The entire process of the selection is very similar to a bin packing algorithm, since we have a target shed for a given time period (TargetBox) and a number of shed offerings from the clients available in different time periods. To achieve the goal, our solution targets a slightly higher shed target. Moreover, the selection algorithm considers the feedback provided by the client on the available shed, reported periodically, as well as a reliability factor, that is defined as follows:

$$T_i = \frac{Avg\,(AvailableS\;hed\,)}{Capacity\; - Avg\,(CurrentUsa\;ge)} \tag{5}$$

where Avg(AvailableShed) is the average available shed reported in the past through the feedback mechanism, Capacity is the load capacity of a client reported at the enrollment and Avg(CurrentUsage) is the average reported usage, again reported through the feedback mechanism. The resulting selection algorithm is depicted in Figure 5.

5.3. Scheduling loads

In order to manage the loads, the eDRAS employs Control Logic (CL) in collaboration with the DR Logic (DL), in the case of Thin Clients. In the case where Smart Clients are present, the functionality described below is distributed between the eDRAS and the Smart Client.

As before, DL handles a DR event and uses an algorithm which, based on static, dynamic and historical data, selects the candidate users and, for each candidate user, it selects the candidate loads that may be controlled. Specifically, the static data consists of the users' features that are stored during the configuration phase (e.g., program that the user has enrolled in, constraints related to availability for control, etc.). The dynamic data consists of the features of each specific DR event (e.g., type) and the parameter values of the DR event. Finally, the historical data consists of the number of control commands that have been issued to each user, the average responsiveness to them, opt-outs, etc. The analytic procedure followed by the DL is the following: it creates a sorted list of candidate end users that may participate in this specific event by applying a number of criteria (fairness, responsiveness, cost, etc.). For each candidate user, DL creates a sorted list of candidate loads (devices) that may participate in this event by applying the same criteria. For each load, DL calculates the amount of power (energy) that has to be curtailed during the DR event (goal), based on the currently scheduled operation for the specific load. This amount of curtailed power is sent to the CL, which tries to meet the required goal by rescheduling the load operation as will be described next. If the rescheduling by the CL succeeds, a new operation schedule is stored at the database and the algorithm proceeds to the next load. Otherwise, if the rescheduling was not successful, the current load is characterized as unavailable and the next load is checked. This procedure continues until all the loads of each candidate user have been checked. The same will be applied for each candidate user, until either the goal has been met or all the users have been checked.

Control Logic is the component that, in close cooperation with the DL, tries to meet each goal set by the DL (e.g., curtailment of 1 kW for 2 hours), by calculating a new operation schedule for each load, without violating the constraints that have been imposed by the user, regarding the preferred quality of service (e.g., comfort, finishing time). It achieves this by using static and dynamic data:

- Static data: It consists of data that stored at the database during the configuration phase, e.g., the thermodynamic model of each house, load types (interruptible, dimmable, shiftable), operational parameters of each load (e.g., consumption profile), restrictions related to the operation (minimum time of operation, etc.), etc.

- Dynamic data: It consists of the required value of curtailed power, requested by the BL and the data that change frequently, either received by sensors (e.g., current interior and exterior temperatures), Internet services (e.g., weather predictions) or by end users (e.g., comfort level, time of finishing a task), ToU or RTP prices, DR events, etc.

The algorithm strives to optimize the energy cost of the end users under a ToU and/or RTP pricing scheme as well as during a DR (e.g., BIP) event, based on the user's preferences related to comfort levels. First, the loads are modeled based on their operational features and their control capability: dimmable (e.g., HVAC), shiftable (e.g., washing machine) and interruptible (e.g., water heater). Specifically, the HVAC system is a continuously operating (dimmable) load, which may be controlled through a thermostat which sets the desirable operating temperatures (setpoints). Based on the work presented in (Ha, Ploix, Zamai and Jacomino, 2006), the required power consumption in order to reach the desired temperature (within a time slot of specific duration, e.g. 15 min) depends on the HVAC's electrical characteristics (average power), the heat capacity and the resistance of the indoor environment and the indoor and outdoor temperatures:

$$P = \frac{T_{j+1} - e^{\left(-\frac{\Delta t}{RC}\right)} T_j}{R\left(1 - e^{\left(-\frac{\Delta t}{RC}\right)}\right)} - \frac{1}{R} T_j^{out} \tag{6}$$

where, P is the average power generated by the HVAC system during time period Δt, C is the heat capacity of the heated (cooled) indoor environment, R is the thermal resistance of the environment, T_j is the indoor temperature at time slot j, T_{j+1} is the desired indoor temperature at time slot $j+1$ (setpoint) and T_j^{out} is the outdoor temperature at time slot j. Shiftable loads (e.g., washing machine) constitute a load type, which may be shifted in time but not interrupted when started nor dimmed, since the power demands of any washing program consist of a number of continuous operational phases, each one posing specific power demands. Finally, the interruptible loads (e.g., water heater), constitute a load type, which may be interrupted but not dimmed. The control algorithm calculates a) the setpoints for the dimmable (increasable/reducible) loads (e.g. HVAC), b) the starting times for the shiftable (schedulable) loads (e.g., washing machine), and the c) starting and stop times for the interruptible loads (e.g., water heater) within a predetermined time period, taking into account both the prices per time slot (ToU/RTP) and the user's comfort preferences. The setpoints of the dimmable loads are calculated by an adequately modified Dijkstra's shortest path algorithm, while the operation schedule of both shiftable and interruptible loads comprises the time slots that maximize the objective function (energy cost and comfort). A more extensive description of the load control algorithm can be found in (Antonopoulos, Kapsalis and Hadellis, 2012). A flow chart of the scheduling operations performed by the DL and the CL is presented in Figure 6.

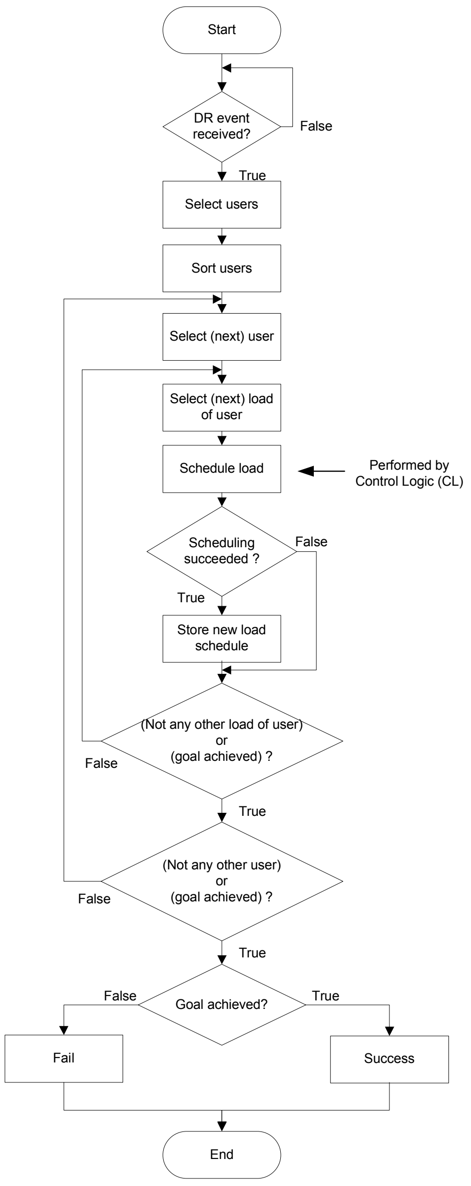

Figure 6: The load scheduling algorithm

6. CONCLUSIONS

In this paper, we have presented an end-to-end DR-based system (DAMAZO) that supports a hierarchical architecture of multiple enhanced DR Automation Servers, according to the emerging business models for the organization of aggregators in Smart Grids. We have based our implementation on the OpenADR v1.0 standard and we have also implemented certain additional functionalities to support the derived complexity of this multi-level architecture. The architectural diagram of our system is provided along with a brief description of the comprising components and the accompanying interfaces.

Future work includes the study of more complex strategies regarding the mapping of different DR programs and the more precise prediction of available shed by clients. Moreover, we will study the outcomes of relevant research projects, (e.g., ADDRESS, SmartHouse/SmartGrid, etc.) to identify similarities and differences and examine whether we can expand our implementation to support the outcomes of these projects. Finally, since the OpenADR v2.0 specification (still in draft state) also provides the communication interfaces for supporting a multi-level hierarchical organization of aggregators, we are going to update our implementation according to the new specification, while preserving the old interfaces so as to offer backward compatibility.

ACKNOWLEDGMENTS
This work has been partially supported by the Greek General Secretariat for Research and Technology (GSRT) as part of the research project "DAMAZO – Automated Demand Response Management System for the Rational Use of Electrical Energy" 09ΣΥΝ-32-613, under the "Cooperation 2009" framework.

REFERENCES

Antonopoulos Ch., Kapsalis V., Hadellis L., "Optimal Scheduling of Smart Homes' Appliances for the Minimization of Energy Cost under Dynamic Pricing", in *17th IEEE International Conference on Emerging Technologies & Factory Automation* (ETFA 2012), Krakow, Poland, September 2012.

BeyWatch, "Deliverable D2.1: Service Requirement specification", Online: http://www.beywatch.eu.

Ha LD, Ploix S., Zamai E., Jacomino M., "A Home Automation System to Improve Household Energy Control", *in 12th IFAC Symposium on Information Control Problems in Manufacturing*, 2006.

EPRI. "Concepts to Enable Advancement of Distributed Energy Resources" White Paper on DER. EPRI, Palo Alto, CA: 2010. 1020432.

Gkatzikis, L., Koutsopoulos I., and Salonidis, T. "The role of Aggregators in Smart Grid Demand Response markets", to appear, *IEEE Journal on Selected Areas in Communications*, 2013.

OpenADR Alliance, "OpenADR 2.0 Profile Specification A Profile", v1.0, 2012.

OpenADR Alliance, "OpenADR 2.0 Profile Specification B Profile", v.0.9, Final Draft, 3-18-2013.

OASIS, "Energy Interoperation Version 1.0, 2-18-2012.

Piette, M.A., G. Ghatikar, S. Kiliccote, E. Koch, D. Hennage, P. Palensky, and C. McParland. 2009. Open Automated Demand Response Communications Specification (Version 1.0). California Energy Commission, PIER Program. CEC-500-2009-063 and LBNL-1779E.

Valtorta, Giovanni, et al. "Architecture and functional specifications of distribution and transmission control systems to enable and exploit active demand." *21st International Conference on Electricity Distribution*. CIRED, Frankfurt, 2011.

Warmer C., et al., "Web services for integration of smart houses in the smart grid", in Grid-Interop Forum 2009.