An Agent-based Market Platform for Smart Grids

Steffen Lamparter, Silvio Becher, Jan-Gregor Fischer
Corporate Technology, Siemens AG, Munich
{firstname.lastname}@siemens.com

ABSTRACT

The trend towards renewable, decentralized, and highly fluctuating energy suppliers (e.g. photovoltaic, wind power, CHP) introduces a tremendous burden on the stability of future power grids. By adding sophisticated ICT and intelligent devices, various Smart Grid initiatives work on concepts for intelligent power meters, peak load reductions, efficient balancing mechanisms, etc. As in the Smart Grid scenario data is inherently distributed over different, often non-cooperative parties, mechanisms for efficient coordination of the suppliers, consumers and intermediators is required in order to ensure global functioning of the power grid. In this paper, a highly flexible market platform is introduced for coordinating self-interested energy agents representing power suppliers, customers and prosumers. These energy agents implement a generic bidding strategy that can be governed by local policies. These policies declaratively represent user preferences or constraints of the devices controlled by the agent. Efficient coordination between the agents is realized through a market mechanism that incentivizes the agents to reveal their policies truthfully to the market. By knowing the agent's policies, an efficient solution for the overall system can be determined. As proof of concept implementation the market platform D'ACCORD is presented that supports various market structures ranging from a single local energy exchange to a hierarchical energy market structure (e.g. as proposed in [10]).

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Design, Economics

Keywords

Smart Grid, Autonomic Computing, Electronic Markets

1. INTRODUCTION

In upcoming years, distribution networks (low and medium voltage power grids) in most countries in the world will be connected to an increasing number of decentralized and often fluctuating energy suppliers like wind power, photovoltaic and CHPs (Combined Heat and Power). In addition, the grid will face a highly volatile and hardly predictable demand as more and more machines and vehicles will be powered by electrical energy (e.g. eCars). In order to ensure the efficiency, reliability and security of future power supply, the *Smart Grid* adds an infrastructure for two-way communication among the connected components to the traditional power grid. By facilitating real-time communication between the grid components, the Smart Grid integrates large, centralized generation units and small, decentralized ones, along with consumers, into an overall structure that can be used for balancing the grid. As information such as load predictions, marginal prices, etc. is distributed over the grid components and a central fully informed entity is not available due to natural information asymmetries and selfish participants (suppliers/consumers), today's central approaches to network control are either highly inefficient or even not applicable any more, e.g. direct control of electrical appliances in a private household through the distribution network provider will not be acceptable for the end user.

Efficient grid control thus requires sophisticated coordination mechanism involving end consumers that actively participate in grid control and therefore make a contribution to grid stability and climate protection. Due to the decentralized nature of the coordination problem, agent technologies seem to provide a useful technological toolbox to tackle the challenges that arise with the upcoming Smart Grid. In this paper, we particularly investigate the applicability of market-based coordination of supply and demand. In many domains markets have proven to be a suitable mechanism for coordinating selfish agents [6, 19] and agent-based energy markets might have substantial impact on improving the efficiency and stability of the future power grid [24, 10, 21].

However, the practical applicability of agent-based energy markets is obstructed by the complexity and diversity of the individual strategies required for participating in the market. Each appliance and generator has different features and parameters, each consumer and provider has different preferences about how to control the devices, various different context and environment information has to be considered, complex forecasting methods are required, etc. All these aspects have to be considered when specifying an agent's bidding strategy. As a special implementation for each strategy has proven to be highly inefficient, this paper presents a generic strategy framework that supports developers to specify agent strategies in a highly flexible way. It can be used

Cite as: An Agent-based Market Platform for Smart Grids, Steffen Lamparter, Silvio Becher and Jan-Gregor Fischer, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1689-1696

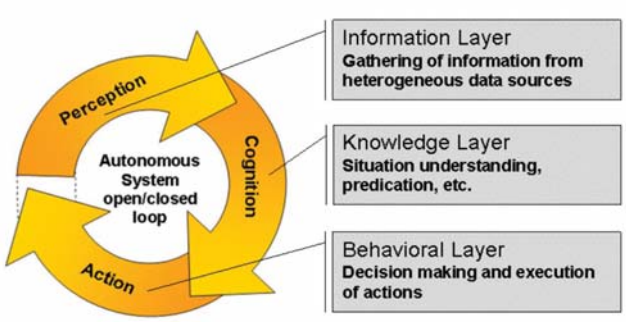

Figure 1: Agent Strategy Framework

to implement heterogenous autonomous bidding agents that are able to interact with different market mechanisms in various domains. The architecture leverages the idea of *policies* [8] for realizing a high-degree of autonomy while making sure that the agents behave within a predefined action space. As these policies are specified as declarative descriptions they can be added and removed at runtime which allows to adapt the strategies dynamically. For example, in the energy market scenario new appliances in the household may come with their policies how they can be regulated. This policies can be used by the energy trading agent to adapt its strategy to the new setting.

The paper is structured as follows. In Section 2, we first introduce the agent bidding process and the generic agent strategy framework. Subsequently, in Section 3 we describe the market mechanism for balancing energy supply and demand. A realization of the overall architecture including market as well as agent implementation is outlined in Section 4. We review related work in Section 5 and finally conclude in Section 6 with a short outlook.

2. AGENT ARCHITECTURE

In this section, we introduce the agent architecture for automated trading on the energy market. The automation involves autonomous acquisition, storage and processing of information by the agent which is also displayed by the steps perception, cognition and action in Figure 1. As also depicted in the figure, these steps can be assigned to the three layers of agent strategy design as defined in [22]. In the following we discuss each layer (step) in more detail.

2.1 Information Layer

The information layer contains information which an agent $i \in \mathcal{I}$ has gathered from the market, the environment or its own private information at time t_k with $k \in \mathbb{N}$. Much in line with [22], we can define the market and agent state as follows.

Definition 1 (Market State) *A market state captures the public information that is available at a certain point in time t_k. Is is defined via a vector $\theta_{\mathcal{M}}(t_k) = (x, B_{t_k}, price_{t_k}, q_{t_k})$ where x represents the trading object, $price_{t_k}$ the clearing price and q_{t_k} the overall traded quantity at time t_k, and B_{t_k} the orders to buy or sell energy which are present in the order book at time t_k.*[1] *The expressivity of a bid b is defined through the bidding language in the market (c.f. Section 3.1).*

[1]Note that full disclosure of B_{t_k} is only available for markets with public order book. Markets with sealed bids usually publish only the highest bid and ask. We thus define the market state as $p_{\mathcal{M}}(t_k) = (x, ask_{t_k}, bid_{t_k}, price_{t_k}, q_{t_k})$.

Given the set of publicly available information on the market, the agents internal state containing private information such as preferences is defined below.

Definition 2 (Agent State) *An agent i's state at time t_k is defined by the vector $\theta_i(t_k) = (id_i, q_{i,t_k}, v_{i,t_k}, comp_{i,t_k})$ where id_i specifies whether the agent acts as buyer or seller, q_{i,t_k} defines the quantity of energy required (or provided) by the agent at time t_k, v_{i,t_k} the reservation price or marginal costs of an agent, and $comp_i(t_k)$ the computational resources available at the given point in time.*

In addition to the Market and Agent State there might be additional information necessary dependent on the application scenario. For example, in the Smart Grid scenario the state of the power grid could be relevant for determining the bidding strategy. As such additional information is not yet perceived by the agent or market, we add an additional *Environment State* which generically captures application specific information.

Definition 3 (Environment State) *The agent's environment state $\theta_{\mathcal{E}}(t_k)$ captures the values of a set of application-specific variables $(e_{1,t_k}, \ldots, e_{n,t_k})$ over time t_k. The variables are not part of the agent itself nor can they be observed on the market directly. They can be rather perceived by the agent when observing its direct environment.*

Typically, information about environment states is perceived via sensors (e.g. measurement of frequency or voltage in a electrical grid) and is aggregated to a higher level of abstraction that can be interpreted automatically by the agents. In the context of Smart Grid initiatives several innovative tools that can be utilized within the agent reasoning process are proposed to capture the grid status ranging from smart meters to advanced power electronics.

Example

First, we have to adapt the information layer to the smart grid market scenario. This requires to adapt the market state to the market mechanism $\theta_{\mathcal{M}}(t_k) = (x, B_{t_k}, price_{t_k}, q_{t_k})$ defined in Section 3.2. As electricity is a highly homogenous good, the trading object x represents simply electricity according to the IEC Norm 60038:1983 with a predefined set of quality criteria, such as frequency between $50hz$ and a voltage level of $230V$ with a tolerance of $\pm 10V$ for example. As the market mechanism does not reveal the bids of other participants we assume order book $B = \varnothing$. The $price_{t_k}$ is a tuple $(\max(a^l, b^{l+1}), \min(a^{l+1}, b^l))$ representing the bid/ask-spread in the market and q_{t_k} is the overall amount of electricity traded at time t_k measured in kWh.

Second, the agent's private state $\theta_i(t_k) = (id_i, q_{i,t_k}, v_{i,t_k}, comp_{i,t_k})$ is adapted as follows: the agent is either a buyer, seller or prosumer (buyer as well as seller), i.e. $id_i = \{seller, buyer, prosumer\}$, q_{i,t_k} represent the maximum amount of electricity that can be provided/consumed by agent i at time t_k, v_{i,t_k} is the maximum/minimum valuation of a single kWh of electricity, and $comp_{i,t_k}$ is currently not used within the smart grid scenario.

Third, the environment state observable by all agents comprises information about the status of the electricity network that can be measured via sensors, such as frequency e_{f,t_k},

voltage e_{v,t_k}, or current e_{c,t_k} and time t_{k,t_k}. Consequently, $\theta_i(t_k) = (e_{f,t_k}, e_{v,t_k}, e_{c,t_k})$. In addition, specific sensor data might be available to some of the agents which could include the current temperature within a fridge, the current load of a manufacturing machine, etc.

2.2 Knowledge Layer

At this layer previously defined information is combined with user or appliance policies given at design/configuration time. These polices capture general rules that define admissible or forbidden actions, respectively. They thereby constrain the strategy space of an agent in the bidding process. In the following definition, we adopt a rather general approach for defining the strategy space S of a market agent.

Definition 4 (Strategy Space) *The strategy space S_i available to an agent i at time t_k is defined by a cartesian product $S_i(t_k) = M \times \Theta_i(t_k) \times A_i$ covering the agent i's action space A_i, the possible states $\Theta_i(t_k)$ and the market mechanism descriptions M. Consequently, a strategy $s \in S_i$ available to agent i defines which action $a \in A$ should be executed for a given market mechanism $m \in M$ in a given state $(\theta_i(t_k), \theta_{\mathcal{M}}(t_k), \theta_{\mathcal{E}}(t_k)) \in \Theta_i(t_k)$.*

The description of a market mechanism is particularly important if more than one mechanism (e.g. one-sided mechanisms like the english or dutch auctions, or double auctions) should be supported. There are several approaches how market processes can be formalized and described [12, 1, 13]. For example, the Game Description Language GDL [13] formalizes games – which are also general formulation of auction protocols – using Datalog and thereby also formally describes the action space for the agents that can be reused in our strategy definition.

By now we have defined the set of possible agent bidding strategies S_i. However, not all of these strategies are equally desirable or even allowed. Therefore, we allow the specification of policies which define whether a certain action is allowed for a market mechanism in a given state or not. In general, policies can be seen as a set of *constraints* that have to be met by a solution to a certain problem. In literature, solving a problem specified by a set of constraints is denoted as *constraint satisfaction problems (CSP)* [4]. A CSP is described by a set of attribute identifiers L – each representing one aspect of the problem – and the domains of these attributes D. As our goal is to specify constraints over the strategy space we assume $D = S$.

Definition 5 (Constraint Satisfaction Problem) *A CSP within the scope of this paper is a tuple (L, D, Φ), where L represents the involved attributes of the problem, D the domains of these attributes and Φ a set of constraints that defines whether a given configuration $c \in C = D_1 \times \ldots ... \times D_n$ is allowed or not. A constraint $\phi \in \Phi$ consists of a* scope *and a* relation*, i.e. $\phi = (scp, rel)$. The scope scp of a constraint is a k-tuple of attribute labels $(l_1, \ldots, l_k) \in L^+$ and the* relation *rel of a constraint the set of k-tuples defining the allowed attribute values $rel \subseteq D_1 \times \cdots \times D_k$ for the given scope.[2] As enumeration of all possible relations is often not feasible (e.g. for infinite domains) we allow relations to be defined via predicates $p_\phi : D_1 \times \cdots \times D_k \rightarrow rel_\phi$.*

[2]Note that the definition assumes that the constraint is defined on the first k attributes.

A strategy $s \in S$ is evaluated with respect to a k-ary constraint with $scp_\phi = (l_1, \ldots, l_k)$ and $rel_\phi = \{(d^{rel}_{11}, \ldots, d^{rel}_{k1}), \ldots, (d^{rel}_{1q}, \ldots, d^{rel}_{kq})\}$ as defined below:

$$G_\phi(s) = \begin{cases} 1 & \text{iff } \exists j \in [1,q], \forall i \in [1,k] : \text{match}(d^{rel}_{ij}, d^c_i) = \text{true} \\ 0 & \text{else} \end{cases} \quad (1)$$

The Equation 1 is evaluated to 1 for a given constraint ϕ and a given strategy s if there is a tuple in the relation rel_ϕ, for which each attribute value d^{rel}_{ij} matches the corresponding attribute value d^s_i in the configuration. The predicate *match* is used to compare two attribute values. In the most simple case, where attribute values represent "flat" datatypes, such as integers or strings, this could be realized by a simple syntactic comparison, e.g. $\text{match}(d^{rel}_{ij}, d^s_i) = true$ iff $d^{rel}_{ij} = d^s_i$.

In order to judge a strategy $s \in S$ as admissible, Equation 1 has to hold for all constraints $\phi \in \Phi$. This is ensured by the following formula:

$$G_\Phi(s) = \prod_{\phi \in \Phi} G_\phi(s) \quad (2)$$

Based on the evaluation of constraints we are able to define the set of acceptable strategies $\hat{S}_i \subseteq S_i$ for agent i by removing the strategies that violate at least one constraint:

$$\hat{S}_i = \{s \in S_i | G_\Phi(s) = 1\} \quad (3)$$

The set $\hat{S}_i$ is therefore the strategy space that has to be considered in the behavioral layer where the best strategy is selected and executed.

Example

In the following, we give specifically for the Smart Grid scenario some example polices that define how an energy agent should behave. Typically, such policies are defined either by the user and specify her/his preferences or are provided by the appliances/generator vendors and specify the devices' working modes.

Demand Profile: A customer (or prosumer) has to be able to specify his preferences with respect to the electricity demand. Typically, the overall required amount of electricity for a single agent $q^{overall}_{i,t_k}$ is split in an amount $\alpha q^{overall}_{i,t_k}$ essentially required by agent i and the sheddable load $(1-\alpha)q^{overall}_{i,t_k}$ that is negotiable according to the market price. In this context, $\alpha \in [0,1]$ is the share of inflexible demand. Thus, the minimal required load can be expressed by constraining q_{i,t_k} (part of the agent state) using the constraint $\phi_{minQ} = (q_{i,t_k}, rel_{minQ})$ with $rel_{minQ} = \{q | q > \alpha q^{overall}_{i,t_k}\}$.

Appliances Specification: As the share of inflexible and negotiable energy depends on the appliances of the customer (i.e. $q^{overall}_{i,t_k} = q^{App1}_{i,t_k} + q^{App2}_{i,t_k} + \ldots$), the demand profile can be constructed from individual policies coming with the appliances. This also means that ϕ_{minQ} could also be defined for individual appliances separately. In addition, policies can regulate whether an appliance such as a fridge, industrial manufacturing machine, etc. can either (i) reduce total load in time t_k to some extent or (ii) shift load from time t_k with high energy prices to a later point in time $t_{k+\epsilon}$ where energy prices are cheaper (i.e. load shedding). An example for a constraint defining that a certain share of load can be shifted within a certain timeframe ϵ is given the following: $\phi_{shedding} = (\{q^{fridge}_{i,t_k}\}, rel_{shedding})$ with $rel_{shedding} =$

$\{q_{i,t_k} | \sum_{t \in [t_s,t_k]} q_{i,t} > q_i^{fridge} \wedge t_k \leq t_s + \epsilon\}$ stating that aggregated electricity within the period $[t_s, t_s + \epsilon]$ has to be above a given threshold q_i^{fridge}. Note that this policy only defines what load shedding strategies can be implemented with a given device, not how an agent determines the share of energy that is finally shifted. The latter requires complex preference orderings and sophisticated energy price forecasting methods [18, 9]. As discussed in [8, 11], the presented policy framework is able to express such more sophisticated strategies using the concept of utility function policies.

Pricing: While for the inflexible load an unlimited price is offered by the customer, the maximal price for the sheddable load depends on the customer's attitude. A simple strategy to define the extend to which electricity is bought above the $\alpha q_{i,k_t}$-level can be implemented using historical market prices. Let $\eta_i \in \mathbb{R}$ be the price elasticity of a customer with $\eta_i = (\Delta q_i/q_i)/(\Delta p_i/p_i)$. A highly negative η_i represents a "savaholic" whereas a η_i near zero represents a rather convenient customer with a low responsiveness w.r.t. the market price.[3] Therefore, we can define a constraint on the customer's reservation price $\phi_{maxPrice} = (\{v_{i,t_k}\}, rel_{maxPrice})$ with $rel_{maxPrice} = \{v \in \mathbb{R} | v < \eta_i v_{i,t_k}\}$ using price elasticity and the valuation.

Analogously to the policies on customer side, policies for electricity producers could be defined. For example, each type of energy plant such as solar plants, wind turbines, or combined heat and power plants come with common policy sets that regulate whether/how production schedules can be changed dynamically, define the marginal costs, etc. In addition, policies may specify regulatory constraints important for the security of energy supplies or antitrust guidelines.

As policy specifications are purely declarative, policies from different appliances ϕ can be combined to policy sets Φ which are evaluated using Equation 2 leading to the set of possible strategies for a concrete setting. Note that this is a huge advantage of the framework as appliances are constantly added or removed and this should be supported in a plug'n'play fashion.

2.3 Behavioral Layer

The behavioral layer is responsible for deciding on the best action to take at each point in time. In order to select an action, we have to rank the strategies according to preferences of the agent. This can be done by defining a utility function $u_i : \hat{S} \rightarrow \mathbb{R}^+$ over the relevant strategy space $\hat{S}$. Then the best strategy is determined by solving the following maximization problem.

$$s' = \arg\max_{s \in \hat{S}} u_i(s)$$

Given the best strategy, we execute the action a' contained in the tuple s' which typically involves sending one or more bids to the market. Note that the utility function and strategy space typically depends on the application scenario as well as market mechanism used. In this context, the available set of actions A is determined by the bidding language of the market mechanism used. For example, in an one-shot auction only one bid can be sent to the mechanism while in sequential auction protocols more complex actions might be involved. Another important application-dependency is the selection of the best price that should be sent to the market. While for incentive compatible mechanisms bidding the real reservation price v_{i,t_k} is the dominant equilibrium strategy for all agents (independent of the strategy of the other agents), for other market mechanisms this is not the case as strategic over- or underbidding might increase the expected return for individual agents. Due to this scenario-dependency we outline the application of the bidding strategy framework using the concrete Smart Grid scenario in Section 3.

[3]For empirical findings on the price elasticities of real energy consumers refer to [20, 21].

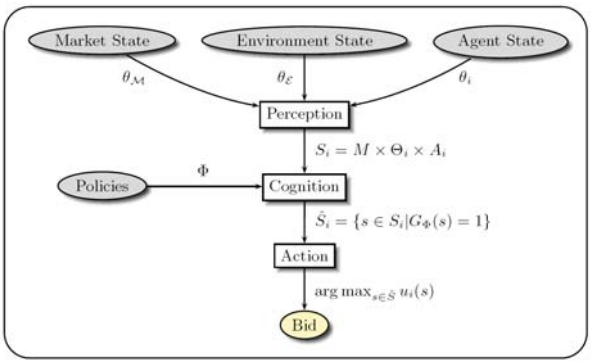

Figure 2: Bidding process according to strategy framework. Note that for sake of readability time-dependency is omitted in the figure.

Example

As we rely on an incentive compatible market mechanism (c.f. Section 3), the behavior layer can be much simplified. In this special case, the action space is simply $A = \mathbb{R} \times \mathbb{R}$ where a tuple represents a bid $b = (v, q)$ defining the maximal valuation of agent v and the required quantity q. As the dominant – and thus u maximizing – strategy of a rational agent is to reveal its true valuation and maximal quantity, the only rational action is to choose strategy $s \in \hat{S}$ with minimal deviation from v_{i,t_k} and q_{i,t_k} defined in θ_i.

2.4 Summary

Before giving some insights into the implementation of the strategy framework we shortly summarize the agent reasoning process as shown in Figure 2. In the first step, market, environment and agent state information are perceived and passed to the cognition step. Here the information is interpreted using the agent's policies. This leads to a set of acceptable strategies that are evaluated using a given utility function. The action contained in the utility maximizing strategy is finally executed and the corresponding bid(s) is/are sent to the market.

3. MARKET MECHANISM

In order to fully specify a market mechanism, we have to define two aspects: a bidding language for communicating the agents' preferences to the market and the mechanism itself consisting of the allocation function $\mathcal{X}$ and pricing function $\mathcal{P}$.

3.1 Bidding language

Generally, a bidding language defines the preferences that an agent wants to reveal to the market, i.e. bidding is about reporting the preference function v_i. When designing a bidding language, there is a trade-off between the expressivity of the language, the privacy loss of users and the complexity of the market mechanism. For example, a bidding language could support expressing how valuation changes depending on time, on available units, etc. For the energy scenario we therefore decided to use a quite restricted bidding language. This has the advantages that we are able to implement a quite efficient mechanism and the agents do not have to reveal too much private data to the mechanism. However, note that this could lead to less efficient markets if dependencies between bids cannot be compensated with local agent intelligence (e.g. smart splitting of originally complex bids into simple bids). A general overview of bidding language with different expressivity can be found in [16].

Based on these considerations, we define the set of *requests* to buy energy B^R and the set of *offers* to sell energy B^O where $B^O \cap B^R = \emptyset$ and $B^O \cup B^R = B$ holds. A *bid* $b_i \in B$ represents a tuple $b = (v_i, q_i)$ where v_i defines the reservation price for a single unit of the good x (i.e. maximal price for requests and minimal prices for offers) and $q_i : X \rightarrow \mathbb{R}^+$ defines how many units of the good are desired/provided. As for the good "energy" it is reasonable to assume divisibility, the overall reservation price is given by $v_i(x)q_i(x)$ or simply $v_i q_i$.

3.2 Mechanism Design

Having defined how agents submit their bids and asks to the market, we are able to define the choice and payment functions. As we have multiple producer and consumer agents in the energy market, our goal in this section is to design a two-sided market mechanism – often called *double auction* or *exchange*. In addition, for energy markets we can assume divisible bids (i.e. partial execution of bids), a call market (i.e. accumulation of bids over a period of time), buy-side and sell-side aggregation of bids, and risk-neutral agents with quasi-linear preferences.

Given the set of requests B^R and offers B^O, the winner determination problem is defined as an allocation function that maximizes the social welfare in the market. The corresponding linear program is defined as follows.

$$\max_{z_{ij}} \quad \sum_{b_i \in B^R} \sum_{b_j \in B^O} (v_i - v_j) q_j z_{ij} \tag{4}$$

$$s.t. \quad \sum_{b_j \in B^O} q_j z_{ij} \leq q_i \qquad \forall b_i \in B^R \tag{5}$$

$$\sum_{b_i \in B^R} z_{ij} \leq 1 \qquad \forall b_j \in B^O \tag{6}$$

$$0 \leq z_{ij} \leq 1 \qquad \forall b_i \in B^R, \forall b_j \in B^O \tag{7}$$

Unfortunately, defining the payment function (and the mechanism as a whole) in a way that the resulting double auction is efficient, incentive compatible and budget balanced is generally impossible as stated by the seminal impossibility theorem of Myerson and Satterthwaite [15]. However, it is possible to design a mechanism that meets two of the three properties. In literature, several different variants have been proposed. Using the well-known Vickrey-Clark-Groves (VCG) mechanism we get an efficient and incentive compatible auction, however, budget-balance cannot be guaranteed any more. To calculate prices the offers B^O have to be arranged in descending order $(b_1, \ldots, b_j, \ldots, b_n)$ and requests B^R in ascending order $(a_1, \ldots, a_i, \ldots, a_m)$ w.r.t. their prices. We then determine the index l where $v_l^b \leq v_l^a$ with v_l^b in b_j and v_l^a in a_i. Given this index l we set the price for buyers to $\max(a^l, b^{l+1})$ and for sellers $\min(a^{l+1}, b^l)$. Other approaches which implement a budget balanced mechanism are – for instance – presented in [14].

4. D'ACCORD - AN AGENT-BASED MARKET PLATFORM

After introducing the conceptual design of the strategy framework, we present the implementation of the framework in this section. In the next section, we give a brief overview of the Market Platform D'ACCORD which also provides a plug-in mechanism and communication infrastructure for agents representing energy consumers and providers. The plugin mechanism is subsequently used in Section 4.2 to implement the bidding process described in Section 2.

4.1 Overview

D'ACCORD is a lightweight distributed negotiation platform providing a .NET-based communication and negotiation infrastructure to software agents that act as participants in negotiations. The agents take an active part in negotiations by following the overall interaction protocol of a market mechanism according to their own local negotiation strategies.

The platform supports arbitrary complex negotiation protocols controlling the overall negotiation process by specifying policies for the initiation and termination of negotiations, participation, submission and if applicable withdrawal of bids. The protocol may include inter alia, known automated negotiation protocols comprising one-to-one (bargaining), one-to-many (e.g. Open-outcry or English auction) and many-to-many relationship such as continuous double auctions [1]. The platform integrates different technologies for computing the negotiation outcomes depending on the requirements of the particular market mechanism. Currently, D'ACCORD supports rule-based execution of configurable negotiation protocols as well as linear optimization for price formation. Figure 3 shows the generic deployment of D'ACCORD runtime nodes which are used to build a flexible market structure.

A running D'ACCORD system comprises at least a single runtime node. However, an arbitrary number of nodes can be supported. The different nodes discover each other using the Peer Name Resolution Protocol (PNRP) or via a central resolver service. Evaluations have shown that for medium-sized distribution networks with about 500 agents and market clearing intervals of 5 seconds a single D'ACCORD node is sufficient. More complex network structures including hierarchical structures – as used by the PowerMatcher system [10] – can be supported.

As shown in Figure 3 agents can either be deployed directly on one of the runtime nodes as .NET dynamic link libraries (plug-in concept) and / or as Web services that implement the interaction and negotiation interface of the D'ACCORD framework. Agents can then participate in one or multiple negotiations that are each dynamically asserted to a runtime node in the D'ACCORD mesh. So-called negotiation hosts manage the context (state space) of one or multiple negotiations and supervise the correct application of the negotiation protocols in a distributed way. The deployment is highly scalable in terms of the number

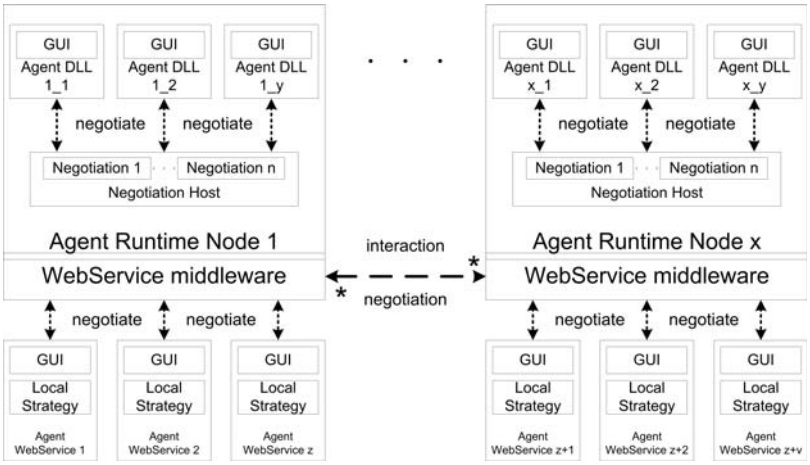

Figure 3: Deployment of D'ACCORD nodes.

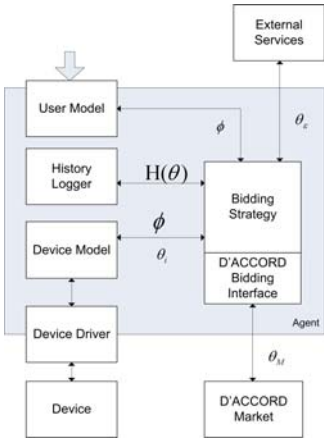

Figure 4: Overview of Agent Implementation.

of D'ACCORD runtime nodes, the number of directly deployed agents and the number of Web service agents that interact with a D'ACCORD runtime node by Web service calls.

4.2 Energy Agent Realization

In this section we present an implementation of the strategy framework presented in Section 2 within a D'ACCORD agent. Figure 4 shows the main component which are described in the following.

Bidding Strategy: This is the main component of the agent and implements the agent reasoning steps perception, cognition and action as outlined in Section 2. Perception at time t_k involves the collection of market state information $\theta_{\mathcal{M}}(t_k)$, agent state information $\theta_i(t_k)$ and environment state information $\theta_{\mathcal{E}}(t_k)$. Furthermore, history state information can be accessed via the History Logger. The state information is then interpreted with respect to the policies Φ acquired from the user model or directly from appliances or generators and the admissible strategies are ranked with the utility function u_i. The evaluation can be done using standard constraint or linear programming solvers. The best bid is sent to the market using the D'ACCORD Bidding Interface. Once an agreement is received the appropriate control commands to the connected devices are issued (e.g. the CHP plant is set to the desired power level).

D'ACCORD Bidding Interface: The D'ACCORD Bidding Interface handles communication with the D'ACCORD Market node. In particular, it provides methods for submitting bids to the market, receiving agreements from the market and receiving further market status information. It currently implements the simple bidding language presented in Section 3.1.

D'ACCORD Market: The D'ACCORD market is represented by a D'ACCORD node which is responsible for the agent. The D'ACCORD node executes the auction mechanism specified in Section 3.2 using a standard linear problem solver.

User Model: On the one hand, the User Model represents the user interface where preferences of the user can be specified. The preferences are expressed via polices Φ and parameters of the utility function u_i. On the other hand, the User Model allows the user to enter electrical devices that cannot be directly accessed (monitored or controlled) by the agent. For example, a user may specify that the agent controls a household with 4 persons. The user may also specify a certain profile (e.g. savoholic vs. comfort profile) that indicates her price elasticity. As long as no smart meter is installed the User Model initializes a device "household" with a standard load profile scaled up to 4 persons.

History Logger: The History Logger is used to store all relevant state information for each point in time t_k.

Device: In this context, a Device represents a (single or aggregated) consumer or producer with metering and often also automated control functionality (e.g. switch off/on, regulate intensity).

Device Driver: The Device Driver is a layer of abstraction between the agent and the hardware devices. Unfortunately, up to now no consensus on a standard communication interface with the different devices has been established within the Smart Grid initiatives. Currently, our solution strives to support at least a part of the IEC 61850 standard with some proprietary extensions. However, depending on the devices that should be connected other communication stacks might

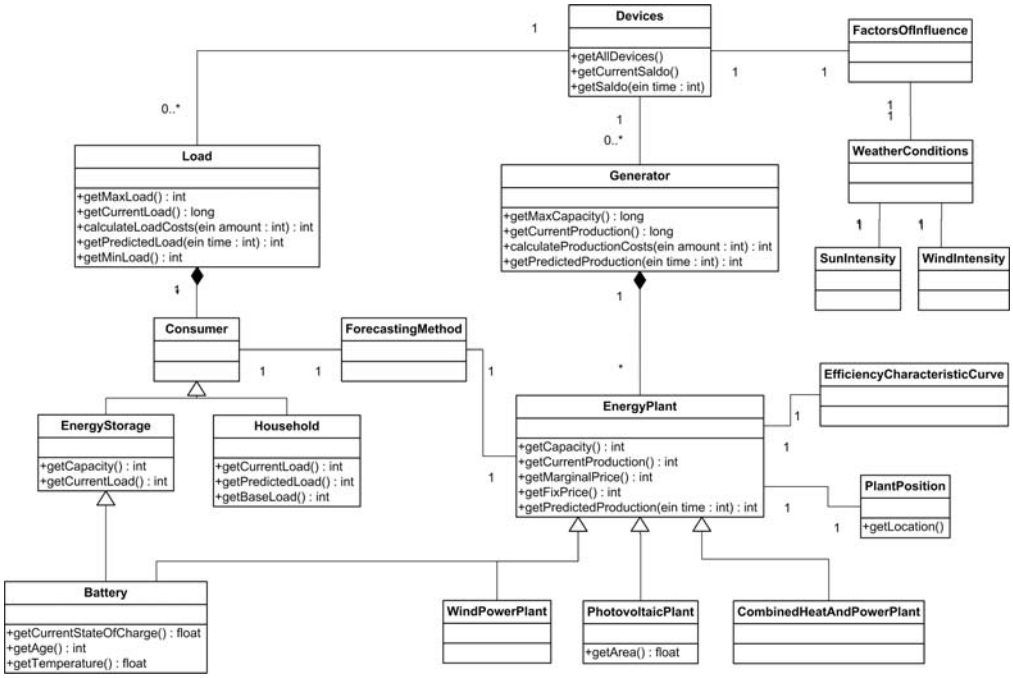

Figure 5: Simplified sketch of the device model.

be required.

Device Model: The Device Model manages all information the agent has about the connected hardware devices. The Device Model allows the Bidding Strategy components to query status information about the device such as maximum/current capacity or current load/production. Moreover, the device model also provides functions for production/consumption forecasting, e.g., for photovoltaic or wind plants based on weather forecasts. Figure 5 sketches parts of the device model. A certain device is represented by a class implementing either Consumer or Energy Plant class (or both in case of a prosumer such as a battery). The model provides methods for deriving aggregated information about all devices, about all Loads, about all Consumers, and individual information about each single device. This is required as user policies may be specified at all levels of abstraction, i.e. policies may regulate parameters of the overall system as well as parameters of a single device.

External Services: The agent architecture has to be able to incorporate external services within the bidding process such as weather forecasting services, services for timeseries analysis, or even external providers of entire strategies for buying/selling energy. In this context, we envision a Web shop that allows energy buyers and consumers to select their preferred services and to combine them easily to implement their own innovative bidding strategies. However, up to now we statically incorporate only a weather forecasting service which is used for predicting future energy production schedules.

5. RELATED WORK

As the focus of this paper is the introduction of a highly flexible strategy framework that can be easily adapted to a wide range of different devices in a plug'n'play fashion, we omit a discussion of simulation results and a performance evaluation of our market-based system at this point. There is a wide range of related work (e.g. [24, 10, 21]) which has shown that – given the right agent strategies – market-based balancing of energy supply and demand could reduce peak loads and increase the efficiency of the power grid compared to traditional power management systems. However, the performance depends purely on the setting (e.g. the devices connected to the grid) and the chosen strategies. Our approach is orthogonal in that we do not discuss the design of individual strategies but provide the framework and infrastructure for expressing the strategies and for automatically applying them to monitor and control a wide range of devices. In the context of designing an infrastructure for market-based energy allocation two major streams of work can be identified: (*i*) On the one hand there are some widely used agent platforms such as JADE (Java Agent DEvelopment Framework) [7] or Cougar [3]. While they provide some basic coordination (including some market) mechanisms such as negotiation and auction protocols, they do not support the agent developers in specifying domain-specific agent strategies for participating in the coordination process. This makes the development of agents often very cumbersome and complicated since for each resource and market mechanism different strategies are needed and no design time support for strategy development is provided. (*ii*) On the other hand, there are a lot of powerful systems implementing (often domain specific) marketplaces, which also provide some means for developing the corresponding agent strategies. The Trading Agent Competition [23] provides a testbed for non-cooperative agent strategies. Commercially highly relevant application examples can be found within the financial domain, where the area of algorithmic trading has become increasingly important over the last years [17]. However, these strategies are specific for a concrete market mechanism and domain. Therefore, they are not

geared towards highly configurable strategies that provide the flexibility to add resources at runtime which is a major requirement in the Smart Grid domain. For example, when developing an agent-based energy market, agents representing households must adapt their strategy in a plug'n'play fashion when adding or removing appliances in the household. While there is work that addresses agent strategy design using more general setting [5, 22], these approaches still lack the flexibility and configurability required to support highly configurable strategies.

6. CONCLUSION AND OUTLOOK

In this paper, we have addressed the complexity problem that obstructs the application of agent-based markets in the Smart Grid domain. By introducing a generic strategy framework that can be governed by declarative policies, the complexity of defining user- and device-specific bidding strategies can be reduced. The reduction of complexity is possible since energy agents implement the same generic framework which can be configured dynamically with policies. As these policies can be provided by device manufacturers, Plug'n'Play for new devices becomes possible.

In the future, we are going to extend our market platform in two directions: First, we are going to add functionality for considering network constraints and transmission losses in the market. This is realized by implementing a nodal pricing approach where the location of a consumer/generator and the real power flow in the grid is explicitly taken into account when determining the allocation and prices. Second, we will extend the agent implementation with more sophisticated methods for strategy design. This involves the development of tools for an easy integration and combination of external services. In order to evaluate our approach more closely we support the idea of an Energy-TAC game as proposed in [2]. In this context, an interesting question is whether all required strategy can be expressed with our policy-based strategy framework.

7. REFERENCES

[1] C. Bartolini, C. Priest, and N. R. Jennings. A software framework for automated negotiation. In R. Choren, A. Garcia, C. Lucena, and A. Ramonovsky, editors, *Software Engineering for Multi-Agent Systems III: Research Issues and Practical Applications*, pages 213–235. Springer Verlag, 2005.

[2] C. Block, J. Collins, W. Ketter, and C. Weinhardt. A multi-agent energy trading competition. ERIM Report Series Research in Management, November 2009.

[3] Cougar – cognitive agent architecture. http://www.cougaar.org/.

[4] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[5] E. Giménez-Funes, L. Godo, J. Rodríguez-Aguilar, and P. Garcia-Calvés. Designing bidding strategies for trading agents in electronic auctions. In *ICMAS '98: Proceedings of the 3rd International Conference on Multi Agent Systems*, page 136, Washington, DC, USA, 1998. IEEE Computer Society.

[6] L. Hurwicz. The Design of Mechanisms for Resource Allocation. *American Economic Review*, 69(2), 1973.

[7] Jade – java agent development framework. http://jade.tilab.com/.

[8] J. O. Kephart and W. E. Walsh. An artificial intelligence perspective on autonomic computing policies. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004.*, pages 3–12, 2004.

[9] W. Ketter, J. Collins, M. Gini, A. Gupta, and P. Schrater. Detecting and forecasting economic regimes in multi-agent automated exchanges. *Decision Support Systems*, 47(4):307–318, 2009.

[10] K. Kok, C. Warmer, and R. Kamphuis. Powermatcher: multiagent control in the electricity infrastructure. In *AAMAS'05: Proc.of the 4th Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 75–82, New York, NY, USA, 2005. ACM Press.

[11] S. Lamparter, A. Ankolekar, D. Oberle, R. Studer, and C. Weinhardt. Semantic Specification and Evaluation of Bids in Web-based Markets. *Electronic Commerce Research and Applications (ECRA)*, 8(1):313–329, Spring 2009.

[12] A. Lomuscio, M. Wooldridge, and N. R. Jennings. A classification scheme for negotiation in electronic commerce. In *Agent Mediated Electronic Commerce, The European AgentLink Perspective.*, pages 19–33, London, UK, 2001. Springer-Verlag.

[13] N. Love, T. Hinrichs, D. Haley, E. Schkufza, and M. Genesereth. General game playing: Game description language specification. Technical Report LG-2006-01, Stanford University, March 2008.

[14] R. P. McAfee. A dominant strategy double auction. *Journal of Economic Theory*, 56(2):434–450, 1992.

[15] R. B. Myerson and M. A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, 1983.

[16] N. Nisan. Bidding and allocation in combinatorial auctions. In *ACM Conference on Electronic Commerce*, pages 1–12, 2000.

[17] D. C. Parkes and B. A. Huberman. Multiagent cooperative search for portfolio selection. *Games and Economic Behavior*, 35:124–165, 2001.

[18] I. Sanchez. Short-term prediction of wind energy production. *International Journal of Forecasting*, 22(1):43–56, January 2006.

[19] T. Sandholm. Distributed rational decision making. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 2000.

[20] A. Siddiqui, E. Bartholomew, and C. Marnay. Empirical analysis of the spot market implications of price-elastic demand. *Berkeley Lab Publications LBNL-56141*, July 2004.

[21] K. Spees and L. Lave. Impacts of responsive load in pjm: Load shifting and real time pricing. *The Energy Journal*, 29(2):101, 122 2008.

[22] P. Vytelingum, R. Dash, M. He, and N. R. Jennings. A framework for designing strategies for trading agents. In *IJCAI Workshop on Trading Agent Design and Analysis*, pages 7–13, 2005.

[23] M. P. Wellman, P. R. Wurman, K. O'Malley, R. Bangera, S. de Lin, D. M. Reeves, and W. E. Walsh. Designing the market game for a trading agent competition. *IEEE Internet Comp.*, 5(2):43–51, 2001.

[24] F. Ygge. *Market-Oriented Programming and its Application to Power Load Management*. PhD thesis, Lund University, 1998.