Electrical Power and Energy Systems 75 (2016) 1–7

Contents lists available at ScienceDirect

Electrical Power and Energy Systems

journal homepage: www.elsevier.com/locate/ijepes

Utilizing capabilities of plug in electric vehicles with a new demand response optimization software framework: Okeanos

Wolfgang Lausenhammer [a], Dominik Engel [a], Robert Green [b,*]

[a] Josef Ressel Center for User-Centric Smart Grid Privacy, Security and Control, Salzburg University of Applied Sciences, Salzburg, Austria
[b] Dept. of Computer Science, Bowling Green State University, Bowling Green, OH 43403, United States

ARTICLE INFO

Article history:
Received 29 September 2014
Received in revised form 14 August 2015
Accepted 21 August 2015
Available online 7 September 2015

Keywords:
Demand response management
Multi-agent systems
Game theory
Plug in electric vehicles

ABSTRACT

Particularly with respect to coordinating power consumption and generation, demand response (DR) is a vital part of the future smart grid. Even though, there are some DR simulation platforms available, none makes use of game theory. This paper proposes Okeanos, a fundamental, game theoretic, Java-based, multi-agent software framework for DR simulation that allows an evaluation of real-world use cases. While initial use cases are based on game theoretic algorithms and focus on consumption devices only, further use cases evaluate the effects of plug in electric vehicles (PEVs). Results with consumers show that the number of involved households does not affect the costs per household. Further evaluation involving PEVs demonstrates that with an increasing penetration of PEVs and feed-in tariffs the costs per household per month decrease.

Introduction

Energy demand in the USA is expected to increase by at least 19%, the supply, in contrast, is only expected to rise by 6% [1]. Furthermore, this energy mismatch is not a US-specific problem [2,3]. While renewable energy could help relieve the load on the grid, it also poses a significant challenge to the grid in terms of keeping supply and demand in balance. With respect to coordination, demand response management (DRM) could pose an ideal solution to this problem [4,5]. DRM refers to "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" [6, 21].

Game theory, in its essence, aims to help understand situations in which several decision-makers interact. Being a mathematical framework and analytical tool, game theory helps study the relationships and actions among rational players. This characteristic renders it an ideal tool to model and understand the inherent complexity of demand response (DR) resulting from this interaction. Publications in this area range from load shifting approaches [7,8] to using storage devices such as PEVs in micro-grid storage games [9] to games that focus on utility companies [10,11]. One thing that these works have in common is a mathematical proof that by optimizing a utility function, a stable point called a Nash equilibrium will be reached [12,13].

This study proposes Okeanos, a novel, game theoretic, Java-based, multi-agent software framework for DR simulation that is capable of investigating the effect of optimizing multiple electric appliances using a game theoretic approach. It is fundamentally different from other DRM software approaches as it plans consumption and production ahead of time. By utilizing game theory, Okeanos benefits from mathematically sound solutions for finding the optimal schedule for household appliances. It supports the simulation of different types of loads and can be configured to work with different game theoretic DRM approaches. The current source has been released as open source[1] and can be used and extended to fit various needs.

While initial results show that savings of up to 6% can be achieved by changing the switch-on time of three household appliances, higher savings can be achieved either by adding more manageable devices to the simulation or by incorporating elective vehicles (EVs) of some sort. In this study, the focus will remain on plugin EVs (PEVs).

The remainder of this paper is structured as follows: The fundamental DR simulation platform, Okeanos, is introduced and key concepts are highlighted in Section "Okeanos"; Results of load

* Corresponding author.
E-mail addresses: wolfgang.lausenhammer@en-trust.at (W. Lausenhammer), dominik.engel@en-trust.at (D. Engel), greenr@bgsu.edu (R. Green).

[1] https://github.com/wolfgang-lausenhammer/Okeanos.

http://dx.doi.org/10.1016/j.ijepes.2015.08.014
0142-0615/

shifting are presented and described in Section "Simulation of multiple households with load-shifting devices"; This is followed by simulations that incorporate PEVs in Section "Evaluation of Okeanos with plug in electric vehicles"; and, finally, Section "Conclusion" concludes this work.

Okeanos

Okeanos is a novel DR simulation platform with a special focus on the inclusion of game theory. Unlike the software presented in [4,14,15], any coordination mechanism that complies with the defined interface is compatible with Okeanos.

Okeanos aims to be a holistic platform for DRM with support for a wide variety of appliances. Through the means of extensibility, new devices can be added by writing a driver for the specific appliance. With OSGi as the foundation, new features can be easily developed, deployed or replaced.

Independent smart household appliances

Similar to other approaches, Okeanos utilizes the multi-agent paradigm to represent household appliances. Thus, with a one-to-one matching between agents and household devices, every device can work towards and set goals or targets on its own. Appliances are proactive and make independent decisions according to the information available to them.

In order not to implement all multi-agent features from scratch, Okeanos builds on JIAC, a feature-rich, modularized and easy to use framework [16]. JIACs modern approach that uses the Spring framework as the basis for the whole system is unique throughout a comparison of multi-agent frameworks including JADE [17], Janus [18] and Jason [19]. Additional evaluation criteria included functionality, active development, ease of use and adoption throughout the software developer community.

In JIAC, the functionality of agents is defined by agent beans. Each bean is a small module with a well-defined responsibility, leading to improved reusability [16]. The energy consumption game described in Section "Coordination mechanism in Okeanos" is an ideal example for this. Its responsibility is to ensure the correct sequential execution of the algorithm. All agents taking part in the schedule optimization process use this bean. Due to the autonomy of agents, it is possible that agents use different games. The meaningfulness of such a mixture, however, is questionable, as no guarantee of the existence of a Nash equilibrium can be given under such circumstances.

The callback functions (cf. Fig. 2) allow for separation of concerns, as the agent itself is still responsible to forward requests to the corresponding components. Similarly, drivers and other services are agent beans as well, ready to be used by agents to support its goals.

Plug in support

OSGi and the Spring framework are two well-known Java frameworks that provide a solid foundation for Okeanos. While both are very powerful tools and offer many features for their respective fields, they share some key concepts, most notably loose coupling and separation of concerns. Naturally, it is beneficial to combine the two and have a module-based, service oriented system as the platform Okeanos runs on, using Spring for the wiring of the components. Eclipse Gemini Blueprint provides a clean and easy to use interface for integrating the two frameworks.

However, to be able to fully utilize benefits of loose coupling, thorough planning is required. Device drivers are the perfect example for the need for extensibility. A flexible and powerful interface eases the interaction with new implementations and the integration of new modules into the system. This is crucial to be able to keep the threshold for developing new modules as low as possible.

With Okeanos built on OSGi, it comprises a conglomerate of various bundles (see Fig. 1) rather than a monolithic core. To allow for optional bundles, the OSGi R5 specification [21] recommends separating interfaces from the implementation in a separate bundle. Consider, for example, a logging service: The application does not necessarily need an implementation for a correct execution, however, at least the interface needs to be present to allow for proper resolution.

As indicated in Fig. 1, every service in Okeanos could be represented in its own module. While, this is possible, it also implies an explosion of projects and, therefore, an increase in complexity. Therefore, layers serve as the boundaries for modules in Okeanos. As recommended, the interfaces of each layer are separated from the implementation and consolidated in different bundles.

Likewise, as it is possible to have no implementation in an OSGi container, it is possible to have multiple implementations present. This is especially true for device drivers, as they all implement the same interface. To be able to distinguish between drivers, additional properties, such as year and brand of a household device, can be specified.

Fig. 1 shows the logic separation between the supporting libraries in the infrastructure bundles area and the application bundles that provide the actual functionality. The Spring extender bundle that is part of the Eclipse Gemini Blueprint project is responsible for activating all Spring powered application bundles and starting up their Spring contexts. This is similar to a J2EE environment, where the Spring application context is started by the application server, whereas here, the extender bundle is responsible for starting all application contexts.

Every such bundle has its own independent context that can import and export services by using special tags[2] in the `context.xml` file. The exported services are regular Spring beans that are registered in the OSGi service registry and, thus, made available to other contexts. For imported services, respectively, Gemini Blueprint searches for a suitable match in the OSGi service registry, fetches it and makes it available to the context.

Coordination mechanism in Okeanos

The requirement of game theory that players have to act rationally is ensured by representing every player by its own agent. Players in this context are household appliances as described earlier.

While there are a number of published game theoretic approaches to DR management [7–11], the game proposed by Mohsenian-Rad and co-authors [7] was modeled with Okeanos as a first proof of concept. Reasons for this include that the algorithm was formulated in pseudo code, which allows for accurate adaptation. Further, potentially more devices can be integrated in the first place by utilizing load shifting as if it were possible with storage devices due to the lack of available data.

The decentralized objective function in [7] with $x^h_{n,a}$ as the energy consumption of a scheduled appliance a of user n at hour h is given by

$$\begin{aligned} \underset{x_n \in \mathscr{X}_n}{\text{minimize}} \quad & \sum_{h=1}^{H} C_h \left(\sum_{a \in \mathscr{A}_n} x^h_{n,a} + \sum_{m \in \mathscr{N} \setminus \{n\}} l^h_m \right) \\ \text{subject to} \quad & l^h_m = \sum_{a \in \mathscr{A}_n} x^h_{m,a} \qquad h \in \mathscr{H} \end{aligned} \tag{1}$$

[2] For detailed instructions on the exact syntax see [22].