From Load Forecasting to Demand Response - A Web of Things Use Case

Yong Ding
TECO, Karlsruhe Institute of Technology (KIT)
ding@teco.edu

Martin A. Neumann
TECO, KIT
mneumann@teco.edu

Ömer Kehri
CAS Software AG
oemer.kehri@cas.de

Geoff Ryder
SAP Palo Alto
geoff.ryder@sap.com

Till Riedel
TECO, KIT
riedel@teco.edu

Michael Beigl
TECO, KIT
michael@teco.edu

ABSTRACT

This paper provides a Web of Things use case from a personalized load forecasting service to a gamified demand response program. Combining real-world measuring applications with web-based applications opens new opportunities to the smart grid. For this purpose, we propose a Web of Things framework for a novel load forecasting process at the appliance level. Firstly, we illustrate the concept design of the Web of Things framework consisting of the sensing infrastructure, the activity recognition and the load forecasting modules. Secondly, we show how we guarantee the modularity and flexibility for implementing all the three modules in a web-based manner. On top of our infrastructure, we propose an extended Web of Things use case by integrating our load forecasting approach into a demand response concept.

1. INTRODUCTION

The fundamental application of the smart grid [18, 16], namely the advanced metering infrastructure (AMI) with the deployment of smart meters, is an enabling factor towards the control of consumer energy demand or loads, known as demand side management (DSM) [7, 19]. Within DSM, mainly two principal activities i.e. load shifting (demand response programs) and load reduction (energy efficiency and conservation programs) can be realized [4].

1.1 Demand Response and the Web of Things

Demand response (DR) programs are designed to shift customers' loads during on-peak periods to off-peak periods in response to time-based rates or other forms of incentives, thereby reducing peak demand, and lowering costs [1, 4]; while energy efficiency (EE) and conservation programs encourage customers to use less power for the same level of end service [4]. DR utilizes usually infrastructure that is already paid for, can therefore be a more cost-effective way than EE or adding generation capacities to meet the peak and stabilize the grid [9]. Most DR programs are currently implemented at the level of end-user customers [1], e.g. residential customers and commercial or office buildings. In such scenarios customer engagement is the core challenge.

Towards this background, in this paper, we present and evaluate a Web of Things (WoT) framework to facilitate the deployment of a DR schema for engaging customers. Based on the work of a resource-oriented WoT architecture from Guinard et al. [10], a vast amount research work (e.g. [10, 12, 14]) was directed towards a common platform for monitoring and controlling interconnected electrical appliances employing web principles. For optimizing power consumption in home or office environments, our previous work [5] proposed an extension of the existing WoT architecture with an additional layer for context-aware applications, where we already sketched the basic ideas underlying this work.

1.2 Load Forecasting as Enabling Factor

In a DR environment, load is not only dependent on traditional influence factors like economic, time and weather factors, and random effects, correlates but also with the information of electricity prices as well as demand response actions [13]. Therefore, an accurate modeling of load behavior of individual consumers due to the customer reaction to the electricity price signals is necessary and needs to be incorporated in load forecasting models [11], in order to achieve an effective DR. With the introduction of real-time pricing (RTP), short-term load forecasting e.g. less than one hour or even five minutes has to be implemented, since DR introduces interactions between demand and electricity price to manage both dispatchable and non-dispatchable DR resources [3].

Load forecasting is therefore of greater importance due to its application in the DR program. Traditional load forecasting in a power system can be classified in three categories by its lead times, namely short-term with intervals ranging from a few minutes to a week, medium-term with intervals ranging from a week to a year, and long-term with intervals longer than a year [8]. Load forecasting can also work at different aggregation levels. System level load forecasting can be used to predict the total load at e.g. the bus or substation levels, where good predictions thanks to repetitive load patterns can usually be achieved [2, 15, 20]. However, at a smart-

meter or appliance level, the closeness to the end consumer implies that load at this level is heavily affected by anomalies like human behavior and activities, which are, by nature, aperiodic, or do not follow predefined patterns [6].

Since DR takes the load information of individual appliances or households into account, for the basic system of the proposed WoT framework, we present in this paper a hybrid approach for load forecasting at the appliance level with varying lead times. It is hypothesized that by sensing and extracting human and group activities in an indoor environment, load forecasting can be improved if the aforementioned consumer behaviors due to human reaction within the DR program are modeled and utilized accordingly. Thus, a complete framework to realize activity-aware load forecasting as the first step of the targeted WoT use case is designed and developed in this work.

2. WOT FORECASTING FRAMEWORK

The most challenging task of such a framework is to utilize information extracted from sensor-based recognition of activities to improve load forecasting. A short analysis of the use case leads to the following requirements:

- There should be an office environment equipped with sensors that continuously monitors the environment.
- Hardware and software infrastructure should be available for reading and collecting sensor data. This component can also be called as the sensing infrastructure.
- An activity recognition component should process sensor readings to extract activity information.
- A load forecasting component should utilize the recognized activity information.

Based on that we defined the system components depicted in Figure 1, in which the arrows between the components represent the information flow and are labeled with the type of information that is exchanged between the components.

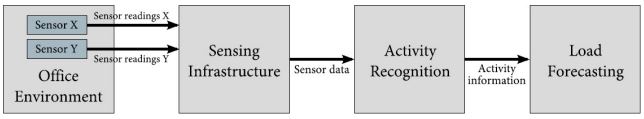

Figure 1: Initial high-level framework design

2.1 Smart Office Environment

The infrastructure of our smart environment consists of heterogeneous wireless sensor networks (WSNs), which includes the following three layers:

Sensing controller: The server software which maintains the sensors and actuators, collects the data from the sensors, and controls the actuators.

Base station: Sensor/actuator controller hardware, which acts as a translator for the communication between heterogeneous sensor networks and the sensing controller. Base stations are device specific, meaning that there can be different types of base stations for different types of sensors.

Sensor nodes: Actual sensor hardware. New types of sensors can be included in the system as long as an appropriate base station is available.

2.2 Sensing Infrastructure

The sensing infrastructure is defined as a component that should receive and store the sensor data from the smart office environment. It should act as a central target for all types of sensors and a central data source for the data processing components that we have on the high-level architecture like the activity recognition and the load forecasting component. In order to manage and store sensor data from different types of sensors in a more flexible and extensible way, we decided to install minimal pre-processing units for every sensor type. These units handle different types of sensor data, which in turn are converted into a uniform data format and sent to the central sensing controller of the infrastructure. Thus, the sensing infrastructure must meet the following requirements:

a) The server software should abstract the underlying data storage engine and offer a standardized REST interface for querying and persisting data.

b) It should be possible to change the data storage engine; therefore the data storage engine should not be accessed directly, only using the interface defined in a).

c) An easier to use web interface should be provided, which is decoupled from the server. The web interface should only use the public REST interface defined in a) and provide real-time monitoring and sensor management.

d) The server software and the web interface should be built in a modular manner to allow the framework integrated into them without much effort.

An overview of the architecture is depicted in Figure 2.

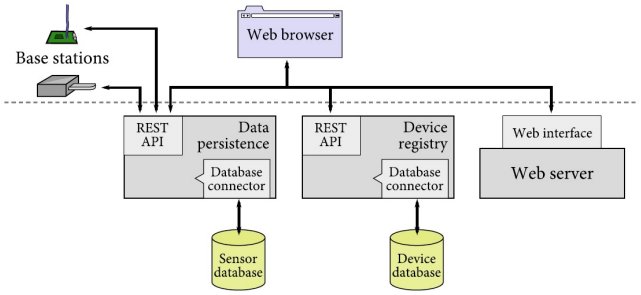

Figure 2: Architecture design of the sensing infrastructure

The infrastructure is then designed to consist of two distinct servers, these being the device metadata registry and the data persistence server. This increases the maintainability of the two entities and also increases the reliability, in a way that a fault in one of the servers cannot disrupt the availability of the other one.

The metadata registry and the data persistence servers have their own data storage engines. To achieve the goal of not being dependent on a single data storage solution, the database accesses in these servers are abstracted, so that only the most generic way of accessing and modifying data is defined as interfaces. There are database connectors which implement these interfaces by using database solution specific commands and behaviors, and to support a new database solution that one should only implement a new database connector which implements the associated database interface.

Furthermore, these two servers are configurable using configuration files which are outside of the application package. The configuration files offer a possibility to choose which implementation should be used for the modular parts of both servers. For example, the desired database connector and the RESTful service endpoints can be configured freely using the configuration files.

The implemented data persistence server transforms the received sensor data in JSON (JavaScript Object Notation) format and uses a data persistence configuration file for the OpenTSDB connector that allows selecting the database engine host of OpenTSDB version 2[1]; while the device registry server manages all sensors and actuators with the internal representation of devices in an entity-relationship model (see Figure 3) and uses a device registry configuration file for selecting the SQLite device database connector to access the SQLite database engine.

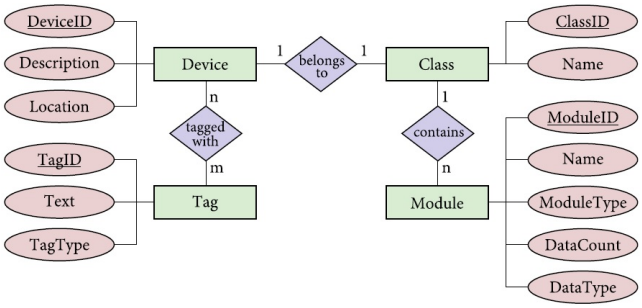

Figure 3: Device registry entity-relationship model

The web interface is completely separated from the server software and hosted on its own web server. Loosely coupling the web interface makes it possible to develop and update it separately from the server software. The web interface is designed as a single-page web application with an object-oriented user interface using HTML and JavaScript. For every entity in the infrastructure, a model object was implemented in a way that the internal data structures of the models resemble the entities which are defined in the infrastructure, so that actual objects can be automatically fetched and modified using the REST interfaces of the device registry and data persistence servers. Furthermore, the various visual representations and interactions are implemented as views, and their appearances are defined and designed using HTML and CSS. Controllers are implemented for managing different parts of the web application which are shown in Figure 4.

[1]An open-source time series database engine offers a powerful REST interface for accessing data: `http://opentsdb.net/`

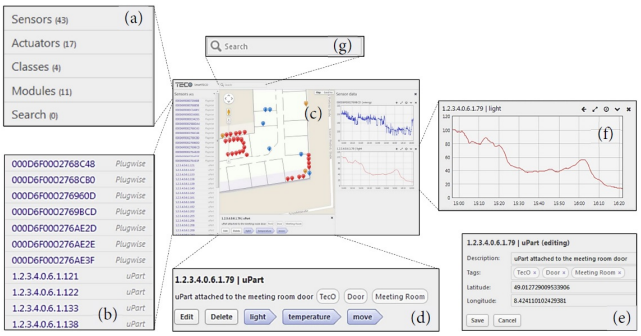

Figure 4: Management web interface showing deployed sensor on floor plan of trial setting

2.3 Activity Recognition

During the research on previous work about activity recognition, it is observed that there are many different methods and approaches that can be used to recognize human and group activities. Therefore, it is decided that the activity recognition methods should be as customizable as possible; meaning any activity recognition method or approach should be easily integrated into the module. The requirements for the activity recognition component can be then summarized as follows: The activity recognition module should be flexible as possible and consist of pluggable "sockets" which represent the different steps of the activity recognition process that should not be hard-coded, and of different "plugs", or interchangeable parts, which represent a method, approach or an extra step whose internals can be defined freely. The only hard-coded functionality should be the data retrieval task from the data persistence server, and the coordination of the various interchangeable parts so that the activity recognition process can run from start to end without user intervention.

To be able to parameterize and set the conditions of the recognition procedure of an activity, the concept of "domain" is introduced. A domain keeps all the needed information for recognizing an activity, like sensors associated with the activity or recognition method, also for making predictions about the future load. A domain is specified by a domain definition file. Figure 5 shows the domain definition file for the domain "meeting" as an example.

The domain definition file is a JSON object, serialized in a file. The individual definitions in the file are described as follow, which bring the activity recognition process in one picture as shown in Figure 6

- First, the name of the domain, in our case, this is the name of an energy relevant activity. There can be different procedures and approaches to recognize an activity, so each one of them should be defined by different domain definition files with a unique domain name.

- The features array lists the sensors that are associated with the activity. These sensors act as data sources, and in the feature generation step, a feature generator of choice will automatically retrieve and process the data from these sensors.

```
{
    "name": "meeting",

    "features": [{
        "name": "projector",
        "id": "000D6F0000D34235",
        "module": "energy"
    }, {
        "name": "door",
        "id": "1.2.3.4.0.6.1.79",
        "module": "light"
    }],

    "features-aux": [{
            "name": "daytime", "column": "daytime"
        }, {
            "name": "calendar", "column": "weekday"
        }, {
            "name": "calendar", "column": "is_holiday"
        }, {
            "name": "calendar", "column": "is_workday"
        }
    ],

    "windowsize": 900,

    "featuregenerator": "generators/generator_std.py",

    "modelbuilder": "models/mb_decisiontree.py"
}
```

Figure 5: A domain definition file example

- The auxiliary features array defines the list of external data that should be included in the feature generation.
- The activity recognition process uses a fixed window size, meaning that the input of the process is sensor data and external data within the interval of the given size, and the resulting activity information belongs to the given time interval. The window size is specified in seconds.
- The feature generator property is used to select the source file which includes the desired implementation for generating features. In our case, the source file is written in Python.
- The model builder property is used to select the source file which includes the implementation for building the model used for recognizing activities. In our case, the source file is also written in Python.

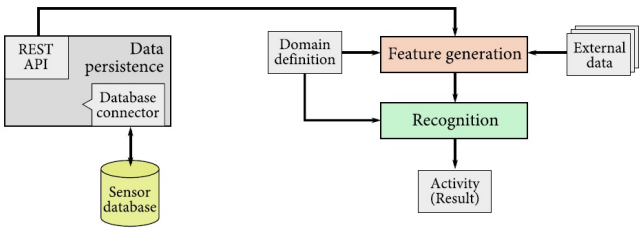

Figure 6: Activity recognition overview in the proposed framework

2.4 Load Forecasting

The load forecasting process is also specified and parameterized in a domain definition file, meaning that for every appliance, a domain should be set up. It is also possible to use the same domain for activity recognition as well as for load forecasting, meaning that using the same domain definition, the activities in this domain can be recognized and the future load of appliances in this domain can be predicted. For this purpose, extra parameters and definitions are appended to the domain definition file, and an example can be seen in Figure 7.

```
"lf-features": [{
    "name": "projector",
    "id": "000D6F0000D34235",
    "module": "energy"
}],

"lf-features-aux": [{
        "name": "daytime", "column": "hour"
    }, {
        "name": "daytime", "column": "daytime"
    }, {
        "name": "calendar", "column": "weekday"
    }, {
        "name": "calendar", "column": "is_holiday"
    }, {
        "name": "calendar", "column": "is_workday"
    }
],

"lf-featuregenerator": "generators/lf_generator_std.py",

"lf-windowsize": 3600,

"lf-previousload": 2
```

Figure 7: Load forecasting parameters in a domain definition file

The feature generator from the activity recognition component is employed and appropriately modified to additionally produce feature vectors based on these parameters for the training phase of the load forecasting process. The output of the feature generator as feature vector consists then of the mean load value from the sensor specified with "lf-features" in the time window t, the external factors specified with "lf-features-aux" in the time window t and the mean load value of the m previous time windows from the same sensor specified with "lf-previousload".

As mentioned before, we aim at a load forecasting process by appending the recognized activity information into the feature vectors, which means that the energy-relevant activity which is supposed to improve the forecast accuracy should be firstly identified. For this purpose, the framework offers the functionality to analyze correlations between activities and mean load values from different domains. The domain for which the load should be predicted is chosen as the source of the load data, and the load data from this domain is analyzed against activities from other domains. For an activity-load pair, the correlation analysis with Pearson correlation coefficient is conducted multiple times by shifting the activity time window as well as by increasing the number of time windows between the activity and the load. For example, the load in the time window t is tested for correlation with the activity in the time window $t-1$, $t-2$, $t-3$ and $t-4$. This process is repeated for the complete data, and the results are evaluated. If a high correlation is found between the load and an activity from another domain with a specific time shift, this activity will be included in the feature vectors used in the load forecasting process.

3. BET AND ENERGY WOT USE CASE

On top of the above described WoT infrastructure, we consider a concrete DR program that was particularly designed to engage users and takes the households' consumption scheduling and the user's behavioral patterns into account.

Bet and Energy[2], is a gamified DR program prototype, which awards electricity consumers remarkably for consuming energy during off-peak time for longer periods. This is implemented using bets that end-use customers can accept as consumption scheduling. In response to the accepted bets, customers will adapt their daily energy consumption, and every won daily bet brings them closer to a monthly goal: they receive the award only when they reach this goal by the end of a month, otherwise they receive nothing.

This differentiates Bet and Energy from other DR programs which are usually based on incremental pricing [1]. This all-or-nothing game shall raise long-term engagement of end-use customers into the DR program to effectively (1) conserve energy, and (2) provide a peak-shaving tool to utilities.

Another characteristic is that Bet and Energy is independent of electricity tariffing of consumers: they may have fixed-rate or dynamic pricing (e.g. day-ahead time-of-use pricing) based tariffs. The goal is to provide a generic tool to any of these tariffs which raises long-term customer engagement.

How would such a betting system be implemented? Utilities or third parties could offer a centralized marketplace for betting on the web, that end-use customers can access using a web browser on their smartphones, tablets, or personal computers. The marketplace would offer bets that ask to restrict the energy usage of a consumer during regional peak-demand times, which is usually in the morning, at lunch time, or at dinner and prime time.

In this scenario, the utility would have to monitor the consumption of end-use customers at relatively high frequency to validate whether they actually have limited their consumption during the betting periods. This could either be implemented by streaming meter data at high frequency to the utility, or, which is more interesting from a data privacy, consumer empowerment and data bandwidth limitation perspective, it could be implemented by placing a program trusted by the utility at the consumer's meter and empowering the consumer to validate the data reported by the program. The meter would then only report won or lost bets to the utility for accounting, whereby maximizing data privacy and consumer empowerment, and minimizing data bandwidth.

The key issue here is how should this be implemented, taking into account that a relationship needs to be established among three different types of devices:

- *Smart meters* trusted by the utility need to continuously measure entire household's consumption for bet evaluation.
- *Servers* of the utility need to trust the bet evaluation results of a program running at customers, e.g. on smart meters or some other permanently active devices.
- *Clients* of the customer (some permanently active monitoring device) need to validate that bet evaluation does not reveal more information, especially metering, than what is necessary for accounting at the utility.

[2]https://www.youtube.com/watch?v=AHcQVuWM5JQ

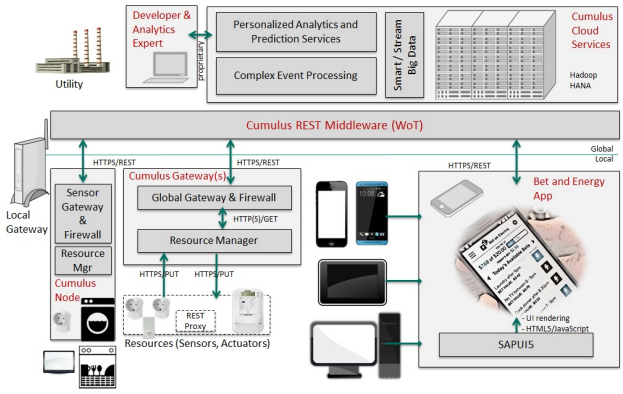

Figure 8: Architecture for integration of Bet and Energy DR program into the existing WoT framework; the server-side big data platform at utility can provide Bet and Energy DR program with personalized analytics and prediction services

In the end, besides smart meters, we need systems to evaluate bets and report to the utility; and we also need systems for monitoring, as seen in Figure 8. This is a client-server oriented DR infrastructure, where the proposed WoT framework comes into play. To implement this ecosystem, an open standard for bet reporting would be key to empower customers to monitor reported data. In addition, to keep cost and development times of systems for monitoring low, the entire communication stack needs to be open and standardized using standards widely adopted on the devices available at consumers, for instance the OpenADR (Open Automated Demand Response) standard [17]. Based on this, monitoring systems implemented entirely in software could be offered quickly and flexibly to the ecosystem of heterogeneous devices at households (windows, android, iphone, ...).

4. CONCLUSIONS

In this paper, we presented a WoT forecasting framework which allowed us to explore the possibilities of integrating energy relevant human and group activity information in an smart office environment into the load forecasting process at the appliance or household level. The goal of this work is to enable new types of applications to engage the customer on different levels. One of the main objectives while designing the underlying framework was not to limit the framework to a limited set of methods and approaches so that the modularity and the flexibility of the framework should be always guaranteed, which positively contributed to the future proof of the framework. The complete system consisting of sensing infrastructure, activity recognition and load forecasting components serves as a basis for real-time execution and evaluation of different activity recognition and load forecasting methods only by implementing the algorithms, leaving

out the need for developing sensor management, data persistence and data retrieval infrastructures as well as activity recognition and load forecasting controllers.

We used Bet and Energy DR web app as a first proof-of-concept application, which awards electricity consumers remarkably for consuming energy during off-peak time for longer periods based on the all-or-nothing game principle. The WoT paradigm however allows us to rapidly prototype new applications and in the mean-time has led to various prototypes ranging from LED displays to web dashboards allowing the assessment of different DR strategies in a living lab environment.

5. ACKNOWLEDGMENTS

This work was funded by the German Federal Ministry of Education and Research (BMBF) as part of the VDAR (grant number 01IS12027) and UHUS (grant number 01IS12051) project. Furthermore, the authors would like to thank the SAP Palo Alto team for their support in prototyping the Bet and Energy App.

6. REFERENCES

[1] M. Albadi and E. El-Saadany. A summary of demand response in electricity markets. *Electric Power Systems Research*, 78(11):1989 – 1996, 2008.

[2] N. Amjady. Short-term bus load forecasting of power systems by a new hybrid method. *Power Systems, IEEE Transactions on*, 22(1):333–341, Feb 2007.

[3] S. Chan, K. M. Tsui, H. C. Wu, Y. Hou, Y.-C. Wu, and F. Wu. Load/price forecasting and managing demand response for smart grids: Methodologies and challenges. *Signal Processing Magazine, IEEE*, 29(5):68–85, Sept 2012.

[4] B. Davito, H. Tai, and R. Uhlaner. The smart grid and the promise of demand-side management. *McKinsey on Smart Grid*, pages 38–44, 2010.

[5] Y. Ding, N. Namatame, T. Riedel, T. Miyaki, and M. Budde. Smartteco: Context-based ambient sensing and monitoring for optimizing energy consumption. In *Proceedings of the 8th ACM international conference adjunct papers on Autonomic Computing*, Karlsruhe, Germany, June 14-18 2011. ACM.

[6] Y. Ding, M. A. Neumann, P. G. Da Silva, and M. Beigl. A framework for short-term activity-aware load forecasting. In *Joint Proceedings of the Workshop on AI Problems and Approaches for Intelligent Environments and Workshop on Semantic Cities*, AIIP '13, pages 23–28, New York, NY, USA, 2013. ACM.

[7] H. Farhangi. The path of the smart grid. *Power and Energy Magazine, IEEE*, 8(1):18–28, January 2010.

[8] E. Feinberg and D. Genethliou. Load forecasting. In J. Chow, F. Wu, and J. Momoh, editors, *Applied Mathematics for Restructured Electric Power Systems*, Power Electronics and Power Systems, pages 269–285. Springer US, 2005.

[9] C. Goldman, M. Reid, R. Levy, and A. Silverstein. Coordination of energy efficiency and demand response. Report lbnl-30443, Lawrence Berkeley National Laboratory, CA, 2010.

[10] D. Guinard, V. Trifa, and E. Wilde. A resource oriented architecture for the web of things. In *Internet of Things (IOT), 2010*, pages 1–8, Nov 2010.

[11] F. Javed, N. Arshad, F. Wallin, I. Vassileva, and E. Dahlquist. Forecasting for demand response in smart grids: An analysis on use of anthropologic and structural data and short term multiple loads forecasting. *Applied Energy*, 96(0):150 – 160, 2012. Smart Grids.

[12] A. Kamilaris and A. Pitsillides. Exploiting demand response in web-based energy-aware smart homes. In *Proceedings of the First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (Energy 2011)*, Venice, Italy, May 2011.

[13] P. Luh, L. Michel, P. Friedland, C. Guan, and Y. Wang. Load forecasting and demand response. In *Power and Energy Society General Meeting, 2010 IEEE*, pages 1–3, July 2010.

[14] N. Namatame, Y. Ding, T. Riedel, H. Tokuda, T. Miyaki, and M. Beigl. A distributed resource management architecture for interconnecting web-of-things using ubox. In *Proceedings of the Second International Workshop on Web of Things*, WoT'11, pages 4:1–4:6, New York, NY, USA, 2011. ACM.

[15] K. Nose-Filho, A. Lotufo, and C. Minussi. Short-term multinodal load forecasting using a modified general regression neural network. *Power Delivery, IEEE Transactions on*, 26(4):2862–2869, Oct 2011.

[16] U. D. of Energy. Title xiii - smart grid sec. 1301- 1308 statement of policy on modernization of electricity grid. Available online, 2007. Energy Independence and Security Act.

[17] P. Palensky and D. Dietrich. Demand side management: Demand response, intelligent energy systems, and smart loads. *Industrial Informatics, IEEE Transactions on*, 7(3):381–388, Aug 2011.

[18] S. G. E. T. Platform. Smartgrids - strategic deployment document for european electricity networks of the future. Available online, April 2010.

[19] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. Jennings. Agent-based control for decentralised demand side management in the smart grid. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, pages 5–12. ACM, 2011.

[20] X. Sun, P. Luh, L. Michel, S. Corbo, K. Cheung, W. Guan, and K. Chung. An efficient approach for short-term substation load forecasting. In *Power and Energy Society General Meeting (PES), 2013 IEEE*, pages 1–5, July 2013.