An application of learning agents to smart energy domains

Alba Amato, Marco Scialdone, Salvatore Venticinque
Department of Industrial and Information Engineering
Second University of Naples - Aversa, Italy
Email: {alba.amato,marco.scialdone,salvatore.venticinque}@unina2.it

***Abstract*—The main requirement for building an Internet of Things is the definition of smart objects in which it needs to put intelligence. The pervasive deployment of smart objects will add value to applications by capabilities of communication, negotiation, learning and distributed reasoning. In this paper we investigate how the paradigm shift from objects to agents is the driver for developing these capabilities by a case study in the context of Smart Energy application domain. In fact the paradigm shift we are seeing in these years is to consider the electricity network like an Internet of Energy, where each and every electrical device and generator will be connected in a network and able to communicate data and receive and react in real time to events and stimuli that arrive from other devices or from the grid: a scattered network of sensors, actuators, communication nodes, systems control and monitoring. Here we present the learning-based approach for power management in smart grids providing an agent-oriented modeling of the energy market. The main issue we focus on is a reasonable compromise between the resolution of the consuming profile representation and the performance and real time requirements of the system.**

***Index Terms*—Intelligent Agents; Smart Grid; Learning**

I. INTRODUCTION

The Internet of Things (IoT) aims at controlling the physical world from a distance but it requires integration and collaboration of different technologies in wireless and wired networks, heterogeneous device platforms and application-specific software. The IoT refers to a globally connected, highly dynamic and interactive network of physical and virtual devices [1].

Similarly Object-oriented programming (OOP), is a programming paradigm based on the concept of objects, which abstracts entities in terms of status and provided services by attributes and methods. By a bottom up approach abstract entities (objects) are identified as parts of the system. Their properties and the interrelationships between computer programs are designed by making them out of objects that interact with one another.

Despite of these similarities OOP just breaks software and information into functional units. This programming paradigm does not deal with providing smartness to objects for enabling intelligent interaction models [2]. The building block of the IoT is the smart object and the novelty is the pervasive deployment of such embedded systems connected to the Internet, interacting with one another. In fact smart objects are mostly fully functional on their own, but value is added by capabilities of communication, negotiation, learning and distributed reasoning.

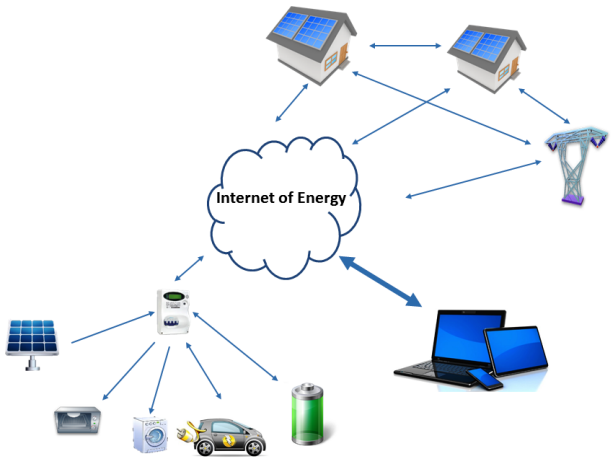

Fig. 1. Internet of Energy Overview

The rapidly growing trend of introducing computing capabilities into everyday objects and places allow us to investigate how the paradigm shifts from objects to agents is the driver for exploiting these capabilities by a case study in the context of Smart Energy application domain.

We are seeing in these years a new generation of electricity network, which behaves like an Internet of Energy, where each and every electrical device and generator will be connected in a network and able to communicate data and receive and react in real time to events and stimuli that arrive from other devices or from the grid: a scattered network of sensors, actuators, communication nodes, systems control and monitoring. In fact the power grid today is no longer designed as a simple network that delivers energy according to one direction: from few large power generation to many small consumption points at the end users.

In particular we address the problem of exploiting the increasing availability of distributed renewable energy sources such as photo-voltaic (PV) panels to improve the energy efficiency in a neighborhood.

In fact, the increasing decentralized production of green energy by photo-voltaic panels has changed the classical model of power supply from the grid to the households.

Nowadays each house becomes a micro-grid connected to the network where energy producers and consumer exchange energy between themselves and with the grid as shown in Figure 1.

As we all know, the electricity demand varies during the day and over the seasons. The electrical current is designed to withstand the maximum level of demand, and then in the absence of constant peaks turns out to be over-sized and underutilized.

The main challenge here is to provide intelligence to consuming and producing devices like photo-voltaic panels, energy storages, appliances, sensors and actuators to let them collaborate in order to agree on the best schedule of energy consumptions according to monitoring information and predictions.

The object oriented paradigm appears to be very useful to represent and manage this type of problem but it is not enough because we need to provide the devices with all advanced capabilities to collaborate as intelligent actors of the IoT.

Intelligent agents will act on behalf of devices learning energy requirements and predicting energy availability. The energy profiles will be then used to negotiate energy exchanges according to which the best global schedule, within a neighborhood, will be built.

In this paper we present the learning-based approach for the distributed energy market. The main issue we focus on is a reasonable compromise between the resolution of the profile representation and the performance and real time requirements of the system. We discuss some experimental results obtained running a prototype implementation.

II. RELATED WORK

Several studies have been conducted regarding the application of multi-agent system in the energy management and negotiation. Paper [3] describes an application of MAS for management of distributed energy resources. Through a software simulation authors demonstrate that is possible to apply a distributed coordination approach to coordinate distributed energy systems. In [4] and [5] a MAS, developed in JADE, is presented for generation scheduling of a micro-grid. The architecture provides several types of agents. For example there is the controller agent that is associated to each device that produces energy such as photo-voltaic panel or wind turbine; load controller agent represents corresponding controllable load in the system. Other several studies have been conducted regarding modeling of the loads. Paper [6] presents a new methodology for modeling common electrical loads. Authors derive their methodology empirically by collecting data from a large variety of loads and showing the significant commonalities between them. A large variety of statistical and artificial intelligence techniques have been developed for short-term load forecasting [7]. Some typical approaches are:

- *Similar-load approach*. This approach is based on searching historical data about loads to identify similar characteristics to predict the next load. The forecast can be a linear combination or regression procedure that can include several similar loads.
- *Regression methods*. Regression is the one of most widely used statistical techniques. For electric load forecasting regression methods are usually used to model the relationship of load consumption and other factors such as weather, day type, and customer class.
- *Time series*. Time series methods are based on the assumption that the data have an internal structure, such as autocorrelation, trend, or seasonal variation. Time series forecasting methods detect and explore such a structure.

In this paper we use a similar-load approach for short-term learning.

Our contribution, and in particular the CoSSMic project [8], is going beyond the state of art by using a distributed negotiation among users' devices on real power grids, that to the authors' knowledge has not been implemented before. The framework will be validated on real infrastructures by trials that involve inhabitants of two different European countries (Germany and Italy). Both software and hardware will be integrated and customized ad hoc to be compliant with existing installations. In [9] we presented a Multi Agent System (MAS) for the deployment of producer and consumer agents that will participate in the energy distribution. We defined a virtual Market that supports the energy negotiation based on XMPP[1] protocol. Agents can make calls for proposals, accept offers and negotiate with other agents. Additional details about how network of agents have been exploited in other applications domains are described in [10].

Load forecasting has always been important for planning and operational decision conducted by utility companies[7], with the deregulation of the energy industries, it will be even more important. However it is critical to predict the evolution of the load demand because it depends on human activity and changes over time with cycles that are daily, weekly, seasonal.

We focus here on a learning approach for short-term load forecasting to estimate load flows and to make decisions about task scheduling. The learning approach means each consuming device behave like an adaptive, intelligent agent, gradually adapting to its environment, and gaining more confidence in its predictions. This has the advantage that great part of the smart-grid configuration is leveraged by agents. The big disadvantage is that the agent has to learn from scratch, which means it might take a some time before the agent can make accurate decisions.

III. THE COSSMIC PROJECT

CoSSMic (Collaborating Smart Solar-powered Micro-grids. FP7-SMART CITIES, 2013) is an ICT European project that aims at fostering a higher rate of self-consumption of decentralised renewable energy production by innovative autonomic systems for the management and control of power micro-grids on users' behalf. This will allow households to optimise consumption and power sales to the network by

[1]Extensible Messaging and Presence Protocol

a collaborative strategy within a neighborhood. In fact the increasing of the decentralized production of green energy is affecting the current energy management scheme both at the grid and at the user level. The amount of electricity used and the energy produced by photovoltaic systems varies over the course of the day, from season to season and depending on the weather conditions. On the other hand the users may not have the opportunity to turn on their appliances when the PV is producing because they do not know when or because they are outside or simply because they do not need those appliances at that time. For this reason there is not an alignment during the time between production and consumption. At the user level this limits both the optimal utilization of green energy and affects the budget spent.

The CoSSMic project proposes an intelligent scheduling of user's consumptions within a micro-grid. A micro-grid is typically confined to a smart home or an office building, and embeds local generation and storage of solar power, and a number of power consuming devices. The main goal is to enable the collaboration among households so that the energy produced by the PVs is consumed by any available consumer in the neighborhood. An autonomic system will able to shift the loads in each neighborhood, according to users' constraints and preferences, in order to find an optimal match between consumption and production during the day, so that the use of renewable energy is maximized. Appliances (refrigerators, washing machine, drier and dishwasher, water heaters, air conditioners) equipped with intelligent controllers are represented by software agents negotiating energy exchanges according to availability, demand, user's preferences and constraints. The energy negotiation between agents will be based on rewards for local producers. To obtain the maximization of self consumption, a predictive approach can be considered, but we need to predict PV production based on the weather forecast and the parameters of the plant. Moreover the power consumed by each device must be known in advance or learned from monitoring information. Also the behaviour of consuming devices can change each day according to a number of parameters (e.g. the external temperature for the refrigerator or the air conditioner, the user's needs for the electric cars or for the oven), which are often unpredictable.

IV. Overview of the CoSSMic solution

Figure 2 shows the main components of our architecture playing the CoSSMic scenario. A detailed description is provided in [9].

The *Graphical User Interface (GUI)* supports interactive control and configuration of the system. It allows to plan the usage of appliances defining constraints and preferences. It also provides to the user real time monitoring information, statistics on historical data and predictions.

Mediator Services are used for storing and management of smart grid information. Mediator services are accessed by device drivers to store measures and by agents to collect information about energy production and consumption of the household. The same services are used to save data about the

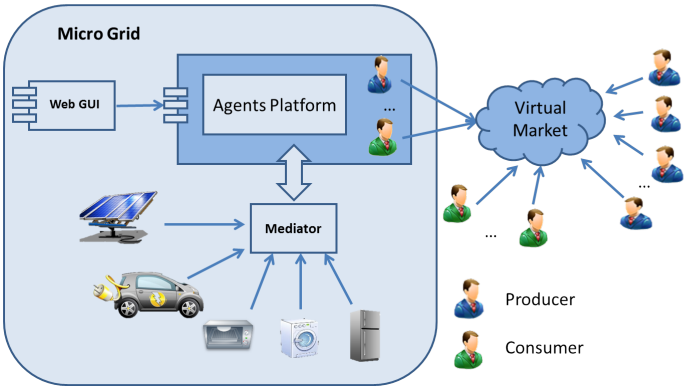

Fig. 2. CoSSMic Platform - Components View

schedule of local devices. In fact the allocation of energy to devices is modeled as task schedule with energy and time constraints.

The algorithm to find the best scheduling of the consuming appliances is designed and implemented as a distributed negotiation among software agents.

Agents are classified according to two categories:

- Consumers: they buy energy for consuming devices. E.g. they will run in houses to manage objects that absorb energy: electric car, computers, ovens, washing machines, etc.
- Producers: they can sell energy. In this category there are, for example, power generators, solar panels, wind turbines.

Those devices, which are able both to produce and consume energy such as energy storages or electric cars, will be represented by a couple of agents belonging to the two different classes defined above.

For the energy negotiation there is not a concrete marketplace, but a virtual market is implemented by a negotiation protocol that uses P2P overlay of agents.

The user will define preferences and constraints about the utilization of his appliances. This policy will be used by the smart devices to find the best plan that maximize the energy self-consumption of the neighborhood. Besides the user's constraint and preferences each agent should know the energy/power profile of its device.

A requirement for producer agents is the knowledge of the energy availability in the future. The prediction about how much energy will be produced by PV panels is computed using weather forecast, properties of the PV plants and historical series.

A requirement for the consumer agents is the knowledge of the consuming profile of the managed device. Of course such a profile will depend on different parameters. In the case of a washing machine the energy consumed could depend on the operation mode, on the amount of clothes, but in the case of an air conditioner or of a freezer the temperature is a relevant feature to take into account. In this case monitoring

information will be used to learn the energy profile while the CoSSMic platform is running.

Devices in the home can send information about electricity consumption through wireless interfaces (for example UHF or Zigbee). Mobile devices (e.g. electric cars), instead, send information through the CoSSMic Cloud. In both cases the information, through the Mediator, reach the agent platform. The mediator integrates also a number of device drivers, which allow to send real time commands to electric devices in the smart house. For example, through device drivers the mediator APIs allow to switch on or switch off devices, when it does not violates any constraints, in order to save energy.

V. Profile Learning

In the following subsection we focus on consuming devices. The prediction of energy production from PV panels will use a different approach and is out of the scope.

A. Energy profiles

In CoSSMic optimization of task scheduling use device profiles. Profiles include some meta-data and time series, i.e. series of time value pairs, which describe the cumulative energy consumed or produced when the device is running. Each device may have more profiles. In fact, as we said before dishwashers and washing machines typically have different operation modes, and in that case there will need one device profile for each mode.

Static or synthetic profile can be used just for the first run. However profiles may change run by run, therefore dynamic profiling is needed. In CoSSMic a learning approach is used to improve dynamically the device profile in real time. For instance, the consumption profile of a heater depends on the ambient temperature. Initial measurements for a certain ambient temperature can be used for the first run. However using monitoring information within the current environment and for the current operation mode will be exploited to update the profile dynamically and to improve the prediction for the next schedule.

B. Learning model

Before the start of the trials and until the software is not available, we used monitoring information to test learning algorithm and to evaluate the proposed approach.

In particular during the trial the user will program the device by the CoSSMic interface and making the system aware about the starting time and the operation mode on the next run. In the following example we used raw energy time-series, which have been collected using a PG&E Landys+Gyr smart meter.

The problem here is that time-series include different working modes which are different for profile and duration and it is necessary to identify the different runs. Another problem is that the samples come at a different rate, according to the energy consumed. To identify different runs of a time series of a washing machine, we split the energy time-series using a supervised approach.

Different runs are recognized by looking at sequence of values whose variation for a certain period (e.g. 10 minutes)

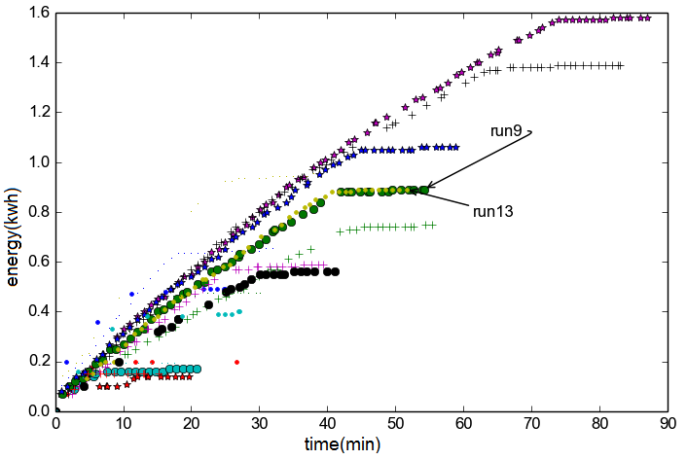

Fig. 3. Cumulative energy of different runs of a washing machine

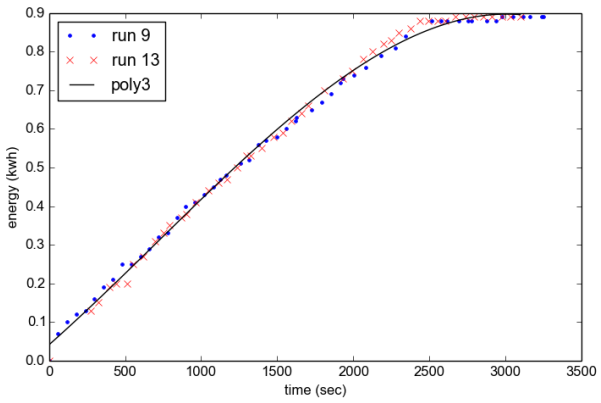

Fig. 4. 3^{th} degree polynomial representation of monitored profile

is below a threshold (e.g. 0.01 kW). In fact in this case we suppose that the washing machine is off and the run is terminated. A new energy increment above the threshold will correspond to the starting time of the next run.

In a second steps we clustered the different runs according to their duration and the amount of consumed energy to identify different operation modes. In Figure 3 the cumulative energy consumption of different runs of a washing machine are shown. The time series run9 and run13 correspond to the same operation mode.

The problem is the best approximation of the profile using two or more time-series by a compact representation that does not affect performance and real time requirements.

In Figure 4 the blue points and the red crosses are the samples from run9 and run13. The black line represent the polynomial 3^{rd} degree polynomial curve that fits the points with a minimal square root error. This representation will introduce of course a residual error, but it is monotone and needs only 5 float parameters to be represented (4 coefficients, duration and residual error), independently from the amount

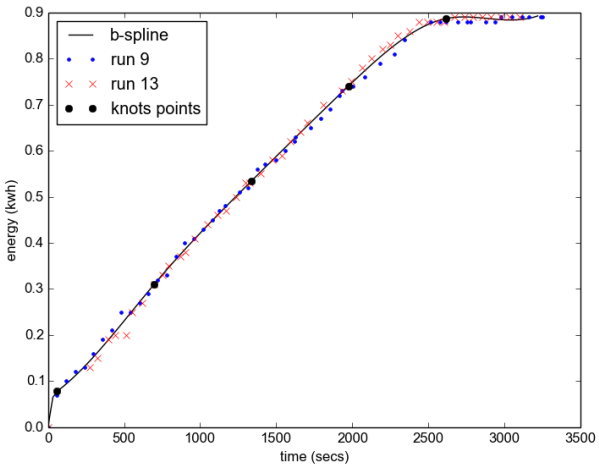

Fig. 5. B-spline representation of monitored profile

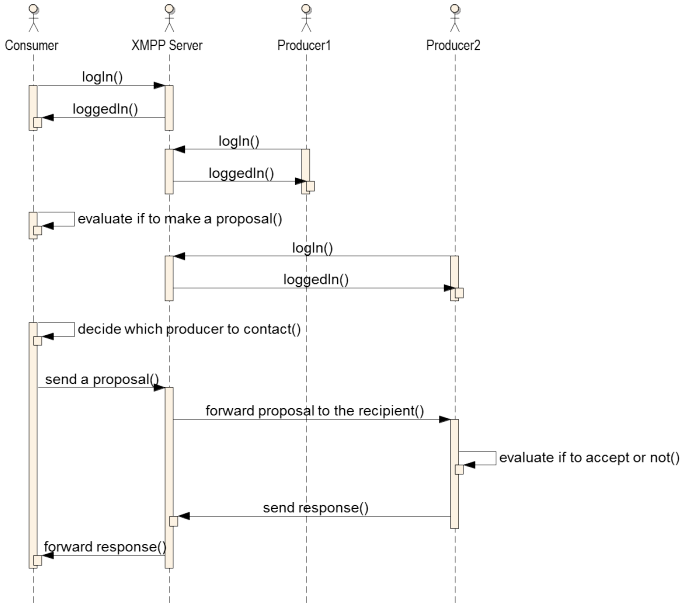

Fig. 6. Negotiation Protocol

raw samples.

In order to improve the resolution of our representation we also considered the utilization of a b-spline. The b-spline is a piecewise polynomial function of degree k. Polynomial curves meet and are continuous in a number of control points. We can control the resolution increasing the number of control points.

In Figure 5 the black line is a b-spline representation of degree k=3 of the best fitting of run9 and run13. In this case 5 control points have been fixed at regular time intervals and the number of float parameters to be transmitted is 27.

VI. SLA BASED ENERGY NEGOTIATION

The main objective of a negotiation is to reach an agreement between a vendor and a customer. A Service Level Agreement (SLA) defines an agreement between a provider and a client in the context of a particular service provision and can be between two (one-to-one) or more (one-to-many or many-to-many) parties.

In our scenario an energy consumer and an energy producer agree to shift the energy workloads of a device to optimise the mapping between production and consumption, but within a time period defined by the earliest start time and the latest start time defined by the user. For this reason the emergent behavior of whole multi agents system will implement a distributed scheduler that allocates consuming and producing tasks for each device within the neighbourhood.

The negotiation protocol will be implemented by an overlay of agents which exchange FIPA[2] messages using XMPP as transport layer. The XMPP server is used for authentication and to support communication across firewalls. Moreover many concepts of the XMPP protocol has been re-used (friendship, presence, multi-user chat, etc..). Alternative server-less solution will be investigated in future works.

[2]Foundation Intelligent Physical Agents

In Figure 6 a sequence diagram about the agents it is shown. A consumer that wants to schedule his consuming tasks will execute the following steps:

- Connects and login to the XMPP server to join the neighbourhood.
- Estimates its own need of energy.
- Brokers the producer that can offer the required energy
- Send a proposal (SLA template)

On the other hand, the producer will:

- Connect and login to the XMPP server
- Wait for incoming proposals
- Evaluate the proposal accepting or not
- Send the related response

Consumers will adopt a ranking mechanism to broker the producer to be contacted. Every consumer associates a rank to each producer, that is an integer that indicates the quality of the producer. Each time the producer accepts the proposal, this number is incremented by 1. The consumer will continue to call the producer which has the highest ranking.

Each producer will try to allocate the consuming workload according to the model presented in the previous section choosing the start time for the incoming task that satisfies the user requirements, but minimizing the use of energy from grid compared to its own production profile.

The negotiation protocol is started by a consumer agent each time a new execution is planned for the handled device sending a proposal. The message body of a proposal is a machine readable SLA template.

Templates used by agents for negotiation define the energy requirements, by the profile discussed above, including user's preferences and constraints (e.g. start date, termination date, cost, etc.). The SLA will complement the energy requirements with the negotiation parties and actual start time.

A producer can eventually withdraw an SLA if its prediction of production change and the negotiation will be restarted by the consumer.

Of course the algorithm may be affected by the problem of local minima that may bring to a not optimal employment of available energy. The final formulation of the optimization algorithm is not available yet, but it is related to a distributed approach where each agent as limited knowledge of the system. We will accept a sub-optimal solution that can be obtained with limited processing and communication resources.

VII. Experiments

After a scouting of available technologies we evaluate performances of SPADE and JADE agent platforms working with different XMPP implementations.

SPADE [11] is an open source agent platform written in Python. SPADE provides a library (SPADE Agent Library) that is a module for the Python programming language for building agents. The library contains a collection of classes, functions and tools for creating agents that can work with the SPADE Agent Platform.

JADE is an agent platform fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications. It also provides a set of graphical tools that supports the debugging and deployment phases [12]. For communication, SPADE is based on the XMPP technology. XMPP [13] is an open, XML-inspired protocol for near-real-time, extensible instant messaging (IM) and presence information. It has also been used for publish-subscribe systems, signaling for VoIP, video, file transfer, gaming, Internet of Things applications. An XMPP server provides various types of services: user account registration, authentication, channel encryption, prevention of address spoofing, message relaying, etc. Nothing prevents to deploy across the network servers that can route and relay messages for workload balancing purpose. SPADE and JADE is fully compliant with FIPA specifications. SPADE use XMPP natively as transport protocol. We developed a plugin for JADE to support XMPP as transport protocol for agent's FIPA-ACL messages .

We experimented three different XMPP server technologies:Tigase, OpenFire and the light implementation provided by SPADE itself. Tigase [14] is an open source project providing XMPP server implementation in Java. OpenFire [15] is an instant messaging and groupchat server that uses XMPP server written in Java and licensed under the Apache License.

The testbed is one host equipped with a 2.67 GHz i5 processor, 4 GB of memory amount and Windows 7 operating system. Here we evaluate the performance running a single consumer and a single producer that always accepts negotiation requests. Message exchanged have an empty payload because the purpose is to evaluate the technology.

In Figure 7 the chart shows the average time to close a transaction. It is clear that with one consumer and one producer, SPADE on OpenFire exhibits the best performance

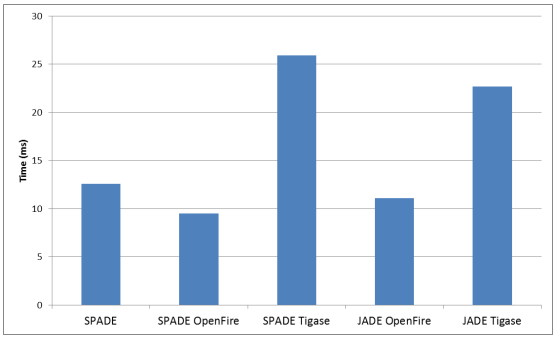

Fig. 7. Performance comparison among SPADE with embedded XMPP server and SPADE and JADE with OpenFire and with Tigase

although the difference with JADE on OpenFire and SPADE with its embedded server is really minimal.

Thanks to this experiment we can affirm that using empty messages the agent platform does not affect performance whereas the time depends on server performance.

In the second experiment we replaced the producer agents that accepted each requests without any computation overhead with a new agent that makes 1.000 floating point operations to simulate the optimization algorithm. We have one producer that always accepts requests and 50 consumers.

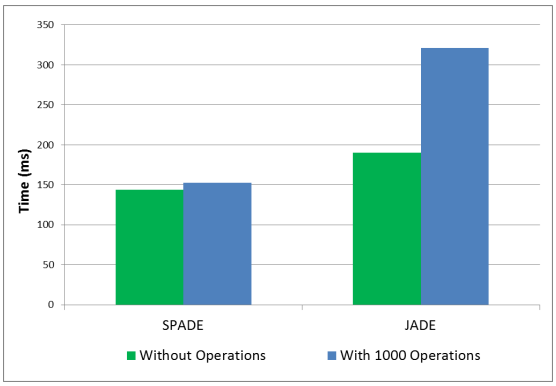

Fig. 8. Comparison between SPADE and JADE with and without operations before response

From figure 8 we noticed that in JADE the time to complete the experiment is greater than SPADE. Moreover, we can note that in SPADE the mean time to close a transaction increases slightly, but in JADE it grows faster. The reason is that Python outperforms Java. As the project trials will use Raspberry Pi to host our software, our choice is SPADE. Because of the limited amount of resources required by its XMPP server we have also decided to host on the Raspberry Pi the SPADE XMPP server. Server to server connection will be used for the communication between agents executing in different households.

In the last experiment we used two RaspberryPi as testbed. It hosts 10 consumers and 1 producer to simulate an household with 10 passive devices and 1 solar panel. Each device is

represented by one SPADE agent. Here we evaluate how the resolution of the profile will affect the performance.

Consumer agents chose randomly the producer to be contacted only the first time whereas for the next time each agent, using a learning mechanism, will contact the most reliable producer. In the experiments to implement the learning capacity we use a ranking mechanism. Each agent creates a vector with all active producers and a ranking is associated to each producer. The ranking is an integer initialized to zero incremented by one each time the corresponding producer accepts the proposal. The consumer will contact the producer that has the maximum ranking and it will continue to contact the same producer until a proposal is refused.

In order to simulate a real scenario we introduce a delay between a request and the subsequent that follows a Poisson distribution with an interarrival of 500 ms.

The request message from consumer is divided in two sections: in the first one there are metadata information, in the second one there is the energy profile.

In listing 1 there is a message example where the meaning of the fields is:

- deviceID: an ID the identify univocally a device in the household;
- EST: Earliest Start Time, the minimum time when the device can be started. The time is expressed using the Unix Epoch Time;
- LST: Latest Start Time, the maximum time when the device can be started. The time is expressed using the Unix epoch time;
- execution_type: the type of execution of that particular device;
- mode: the mode of operation of that particular device;
- taskID: the ID of task;
- dataload: three real coefficients and a value that indicated the duration.

Listing 1. Request Message Example

```
{ "metaload":
  [
    {"deviceID":"61"},
    {"EST":"62340"},
    {"execution_type":"single_run"},
    {"LST":"80340"},
    {"mode":"Delicates"},
    {"taskID":"29"}],
  "dataload":
  ["0.1223","0.2342","0.4351","1426607214"]
}
```

The real parameters represent the coefficients of a 3^{th} degree polynomial curve. For lake of space we do not show the results when the raw profile is sent, but we can see that such solution is not feasible with the available resource. The response message contains the Actual Start Time (AST) for the current run, i.e. the time when the producer can provide energy.

We observed that the mean time to obtain the response is of about 5.3 seconds with a standard deviation of 676 milliseconds in the case of 10 consumers and 1 producer for each Raspberry, using a delay between requests and adopting a ranking protocol for brokering.

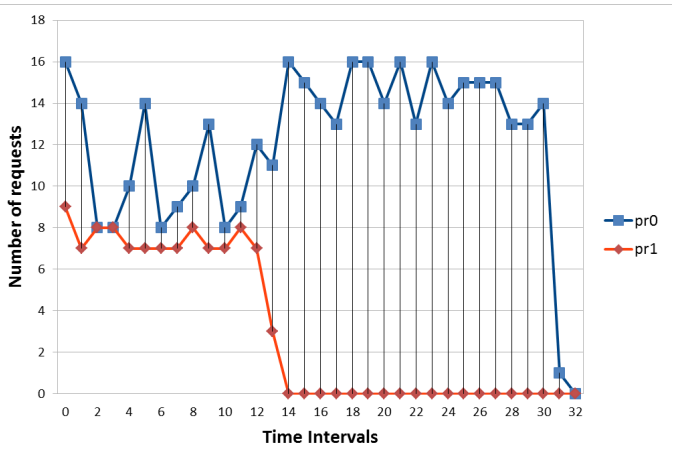

Fig. 9. Number of Requests in each time interval

The chart in Figure 9 represents the number of simultaneous requests per producer within intervals of 10 seconds. The producers manage an average of 11 requests each 10 seconds, more than 1 every second.

It can be seen that when a producer finished the energy, all the consumers move ask to the other producer. We concluded that number of requests that the system greater that the arrival rate in the real case.

VIII. Conclusion

This paper focused the exploitation of software agents as the building blocks of IoT, whose main challenge is the to design and development of application by the interaction of pervasive smart objects. We presented the approach and activities of the European Project CoSSMic that investigates the optimization of the decentralized energy production from photo-voltaic panels. We introduced the concept and the high level architectural. We focused on the design of smart devices that collaborate to find the best schedule of consumptions by a distributed negotiation. In particular a preliminary investigation about the learning model to predict the energy profiles of consuming devices and performance evaluation of the negotiation prototype have been presented.

Acknowledgments

This work has been supported by CoSSMic (Collaborating Smart Solar powered Micro grids - FP7 SMARTCITIES 2013 - Project ID: 608806).

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] L. Benedicenti, "Rethinking smart objects: building artificial intelligence with objects." *ACM SIGSOFT Software Engineering Notes*, vol. 25, no. 3, p. 59, 2000.

[3] M. S. Narkhede, S. Chatterji, and S. Ghosh, "Multi-agent systems (mas) controlled smart grid a review," *IJCA Proceedings on International Conference on Recent Trends in Engineering and Technology 2013*, vol. ICRTET, no. 4, pp. 12–17, May 2013.

[4] T. Logenthiran, D. Srinivasan, A. Khambadkone, and H. Aung, "Multi-agent system (mas) for short-term generation scheduling of a microgrid," in *Sustainable Energy Technologies (ICSET), 2010 IEEE International Conference on*, Dec 2010, pp. 1–6.

[5] D. Srinivasan, T. Logenthiran, A. Khambadkone, and H. Aung, "Scalable multi-agent system (mas) for operation of a microgrid in islanded mode," in *Power Electronics, Drives and Energy Systems (PEDES) 2010 Power India, 2010 Joint International Conference on*, Dec 2010, pp. 1–6.

[6] S. K. Barker, S. Kalra, D. E. Irwin, and P. J. Shenoy, "Empirical characterization and modeling of electrical loads in smart homes," in *International Green Computing Conference, IGCC 2013, Arlington, VA, USA, June 27-29, 2013, Proceedings*, 2013, pp. 1–10.

[7] E. Feinberg and D. Genethliou, "Load forecasting," in *Applied Mathematics for Restructured Electric Power Systems*, ser. Power Electronics and Power Systems, J. Chow, F. Wu, and J. Momoh, Eds. Springer US, 2005, pp. 269–285.

[8] A. Amato, R. Aversa, B. Di Martino, M. Scialdone, S. Venticinque, S. Hallsteinsen, and G. Horn, "Software agents for collaborating smart solar-powered micro-grids," in *Smart Organizations and Smart Artifacts*, ser. Lecture Notes in Information Systems and Organisation, L. Caporarello, B. Di Martino, and M. Martinez, Eds. Springer International Publishing, 2014, vol. 7, pp. 125–133.

[9] A. Amato, B. Di Martino, M. Scialdone, S. Venticinque, S. O. Hallsteinsen, and S. Jiang, "A distributed system for smart energy negotiation," in *Internet and Distributed Computing Systems - 7th International Conference, IDCS 2014, Calabria, Italy, September 22-24, 2014. Proceedings*, 2014, pp. 422–434.

[10] A. Amato, B. Di Martino, and S. Venticinque, "Semantically augmented exploitation of pervasive environments by intelligent agents," in *ISPA*. IEEE, 2012, pp. 807–814.

[11] M. E. Gregori, J. P. Cámara, and G. A. Bada, "A jabber-based multi-agent system platform," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '06. New York, NY, USA: ACM, 2006, pp. 1282–1284.

[12] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "Jade: a software framework for developing multi-agent applications. lessons learned," *Information and Software Technology Journal*, vol. 50, pp. 10–21, 2008.

[13] P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Core," Internet Engineering Task Force (IETF), Tech. Rep., 2011.

[14] T. Inc., "Tigase XMPP server," http://projects.tigase.org/projects/tigase-server.

[15] I. Realtime, "OpenFire XMPP server," http://www.igniterealtime.org/projects/openfire/.