Using intelligent agents for self-adaptation and self-optimization of energy consumption in power networks

Evangelos Pournaras Martijn Warnier Frances Brazier
Intelligent Interactive Distributed Systems
VU University, Amsterdam
The Netherlands
{E.Pournaras,M.Warnier,FMT.Brazier}@few.vu.nl

Abstract

Global stabilization of energy networks without centralized control is the challenge this paper addresses. An agent-based approach to decentralized self-management of networked appliances is the solution this paper explores. Software agents represent thermostatically controlled appliances (TCAs), generate energy plans for expected energy consumption, and interact with each other as peers within a tree based peer-to-peer overlay. The Energy Plan Overlay Summation (EPOS) mechanism proposed, propagates plans generated by individual TCA agents to aggregators within this structure to achieve self-optimization/stabilization of energy requests. Preliminary results in a small-scale and restrictive environment are promising: a 15% increase in energy stabilization is achieved.

1. Introduction

Power management in fully decentralized systems is a challenge for distributed systems and autonomic computing [1]. As variance in power consumption can be costly, research addresses fluctuations in demand over time [2], and adapting production accordingly. This paper focuses on decreasing fluctuations in demand of networked *thermostatically controlled appliances* (TCAs [8]), by aggregation of expected energy requirements over time. Minimizing fluctuations of energy consumption by such devices could have a crucial effect on global energy production and consumption as these devices are currently said to consume 25% of the energy use in the U.S.A. [9].

Decentralized power management, however, requires intelligence within a networked system. Software agents are the means chosen to provide this intelligence, as also chosen by e.g. [3, 4, 5]. TCAs (refrigerators, air conditioners, water heaters, freezers) equipped with intelligent circuit controllers [4] are represented by software agents. Each individual software agent, representing an appliance, has knowledge of its energy needs, of the potential requests to be made over time by its environment. On the basis of this knowledge and the environment in which it is situated each agent devises its own energy plans.

As large-scale systems with centralized aggregators suffer from extreme processing and communication cost and scalability problems [10], decentralized aggregation of energy plans is the approach this paper proposes. Different types of aggregation are explored: summation being the simplest, more complex aggregation operations gather, coordinate and selects the best plan for each agent on the basis of more extensive knowledge. The best plan is the one that maximizes and contributes best in the stabilization of global energy consumption, given an agent's knowledge.

To structure communication and aggregation a tree based peer-to-peer overlay structure is assumed, in which agents have the dual role of requester and aggregator[1].

Aggregation over an overlay benefits from a minimum communication cost as the hierarchy imposes one message per node in every aggregation round, compared to other solutions such as, for example, gossiping protocols [6, 7] that exchange messages continuously over time.

The *Energy Plan Overlay Summation* (EPOS) mechanism proposed, deploys the hierarchy in the network and the individual intelligence of TCAs to aggregate plans, locally selecting the best plan to satisfy the global goal of energy stabilization over time.

2. Aggregation in EPOS

EPOS is a self-adaptation mechanism developed to aggregate energy plans devised by TCA software agents, to

[1]Note that the authors are aware of the fact that trees are sensitive to failures, but are also aware of existing solutions that can build and maintain robust trees resilient to failures [12].

minimize fluctuations in demand, acquiring stability, minimizing oscillations in a global energy network. Self-optimization emerges through coordination of plans by agents. This section outlines the functionality of EPOS: self-adaptation by plan aggregation and selection and global self-optimization by aggregation within a hierarchical overlay structure.

2.1. Aggregation Environment Overview

The EPOS aggregation environment is based on the three following concepts: *software agents* that represent the TCAs, *energy plans* that describe the possible options for energy consumption over time, given the TCAs needs and characteristics, and a *peer-to-peer tree overlay* with its nodes being the aforementioned agents. Each agent generates a number of energy consumption plans for the expected energy requirements for a given period of time. Such plans are functions that contain power consumption values for a number of discrete time intervals. Agents are organized in a peer-to-peer based hierarchical tree overlay for communication, plan aggregation, matchmaking and selection. The hierarchical structure itself is currently arbitrary. Agents choose the most appropriate plan for their own TCA for a following time period, given the knowledge they have of their own options, as a requester, and aggregate their children's plans.

EPOS exploits the arbitrary tree based overlay structure to coordinate plans. Each agent in a tree sends a set of its own potential plans together with the old and new cumulative plans that have been calculated so far to its parent. The latter gathers these plans, pre-processes them by creating all possible combinations of possible plans and adapts them in a locally optimum plan. This allows the parent-aggregator to choose the more stabilized plan. The new overall "global plan" is calculated gradually during an aggregation round. The procedure is recursive, starting from leaves up to the root. In each step, the calculations are performed at the level of the tree at which agents are aggregators and their children requesters. The cumulative plans are updated and, in the final step, the self-adaptive plans converge to a "new globally optimum plan". During aggregation, agents retain some memory of the previous plans selected and of the global energy consumption plan that is broadcasted back to all the agents of the tree to initiate the next aggregation round. This memory is referred to as *overlay memory* and its role is outlined in the following sections.

In the remainder of this paper, a tree T is defined as a list with each of its nodes being an agent n. Each agent n has a list C containing its children, with each element representing a child agent c. In T, a set of all of the potential branches can be identified with every branch B defined by its root $b \in T$ and includes all the nodes from b to the leaves. Each agent generates a set of possible plans P_{s} with each element of the set containing a plan p. A plan p is a function $p[t] = e$ that gives the (constant) power consumption e in the time interval t. Any reference of the form m_{i}^{j} describes the metric m with description i, concerning agent j.

2.2. Information Exchange

In each aggregation round, agents exchange information in two directions over the aggregation tree. The bottom-up exchange concerns the aggregation, and it starts from the leaves up to the root. Top-down exchange concerns the broadcast of the global aggregated energy plan estimated as the summation of all of the selected plans of the agents in the tree. This benefits the adaptation in the next rounds, as explained in Section 2.4, and initiates the next aggregation round without additional messages. In every step, the aggregators return the selected plans to their children-requesters. The aggregation is based on the following tuple:

$$< P_{\mathrm{s}}^{c \in C^n}, s_{\mathrm{bo}}^{c \in C^n}, s_{\mathrm{bn}}^{c \in C^n} > \quad (1)$$

where the agent n receives the possible plans P_{s}^{c} of its child $c \in C^n$ and the sum of the old and new selected plans, $s_{\mathrm{bo}}^{c \in C^n}$ and $s_{\mathrm{bn}}^{c \in C^n}$ respectively, as they has been calculated in the branch with c as root. This tuple is sent by the requester to the aggregator (child to parent). The aggregator must receive the tuple for each of its children before it starts the plan pre-processing.

2.3. Plan Pre-Processing and Convergence

Pre-processing concerns the generation of candidate plans from all the possible plans (P_{s}) received from children. The candidate plans are generated by performing parallel summation (summation of lists) over the values of all the distinct plan combinations. In addition, the cumulative summations of old and new plans, s_{bo}^{c} and s_{bn}^{c}, from all the children are summed in order for the aggregation to converge one level up in the tree.

The list with the candidate summation plans of all of the combinations is expressed as S_{c} and its size is given by (2). The cost of the calculations depends on $|C|$. The processing cost per agent can be configured and controlled by adjusting the number of (direct) children in the tree.

$$|S_{\mathrm{c}}| = \prod_{c \in C} |P_{\mathrm{s}}^{c}| \quad (2)$$

In every step, i.e., a level up in the tree, a cumulative summation of the old and new plans in the branches is performed respectively. The procedure can be outlined as a

Table 1. Transitions in the convergence of plan aggregation

Cumulative Plan	Agent (leaves level)	Agent-to-Branch (branch level)	Branch-to-Tree (global plan)
Old	$s_{\text{o}}^n = \sum_{c \in C^n} p_{\text{o}}^c + p_{\text{o}}^n$	$\xrightarrow{s_{\text{o}}^n \xrightarrow{C^n \subseteq B^n} s_{\text{bo}}^b} s_{\text{bo}}^b = \sum_{c \in C^b} s_{\text{bo}}^c + p_{\text{o}}^b$	$\xrightarrow{s_{\text{bo}}^b \xrightarrow{B^n \subseteq T} s_{\text{go}}} s_{\text{go}} = \sum_{n \in T} p_{\text{o}}^n$
New	$s_{\text{n}}^n = \sum_{c \in C^n} p_{\text{n}}^c$	$\xrightarrow{s_{\text{n}}^n \xrightarrow{C^n \subset B^n} s_{\text{bn}}^b} s_{\text{bn}}^b = \sum_{c \in C^b} s_{\text{bn}}^c$	$\xrightarrow{s_{\text{bn}}^b \xrightarrow{B^n \subseteq T} s_{\text{gn}}} s_{\text{gn}} = \sum_{n \in T} p_{\text{n}}^n$

recursive calculation starting from the leaves up to the root of the tree. Table 1 illustrates the evolution of summations over time during an aggregation round. The two long arrows in the formulas denote the conceptual *agent-to-branch-to-tree* transitions. The shorter arrows in each case show how the transition is achieved. Initially, the branches contain the leaves and their parents. The summation is performed on the old and new selected plans, p_{o}^c and p_{n}^c respectively. During the *agent-to-branch* transition the branches grow, becoming higher, but the number of branches becomes lower as the cumulative aggregation plans are summed. The child agents in the first step are a subset of the agents contained in the branches of the next aggregation steps ($C^n \subset B^n$). The operation can also be described and visualized as merging of branches according to Fig. 1 (in the next section). At the end of the aggregation round, the procedure converges with only one branch representing the whole tree. The final summation in the aggregation cycle represents the global sum of plans running. This is the *branch-to-tree* transition and the arrow reflects the convergence of the branches to the tree T ($B^n \subseteq T$).

The summations in the old and new plans are not equivalent. The node n acts as an aggregator and not as a requester in this aggregation step, thus the new cumulative aggregation plan of the branch s_{bn}^n does not contain the selected plan p_{n}^n of agent n.

2.4. Adaptive Global Aggregation Plan Operator

Considering the aforementioned actions, agents retain the following information: The global old plan s_{go} of the previous aggregation round, the new (s_{bn}) and old (s_{bo}) cumulative summation plan of the branch and the candidate summation plans S_{c} of their children. This information is the input of the *Adaptive Global Aggregation Plan Operator* (AGAPO). AGAPO forms the core of self-adaptation towards global self-optimization from an agent viewpoint. The operator adapts the candidate plans to the old global aggregated plan and to the cumulative summation plans in the branch. The overlay memory ($s_{\text{go}} - s_{\text{bo}}$) provides some knowledge about the plans of the upper agents in the tree from which there is no new knowledge. The new knowledge ($s_{\text{bn}} + s_{\text{c}}$) is a subset of the new global plan formed at the end of the running aggregation. The calculation of the AGAPO plan is given below:

$$s_{\text{agapo}} = \underbrace{s_{\text{go}} - s_{\text{bo}}}_{memory(T \setminus B)} + \underbrace{s_{\text{bn}} + s_{\text{c}}}_{new(B)} \tag{3}$$

The AGAPO plans are calculated for every candidate plan ($|S_{\text{agapo}}| = |S_{\text{c}}|$) and the list of these new plans are those that are examined for their stabilization and oscillations. Fig. 1 illustrates the convergence and adaptation of the cumulative plans. The changes in branches and aggregation points are depicted together with the effect and contribution of the overlay memory in every step of an aggregation round. The example in Fig. 1 shows 5 consecutive steps during one aggregation round.

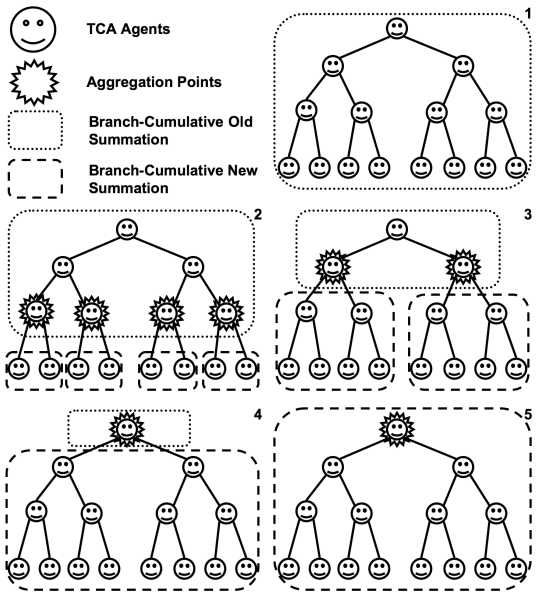

Figure 1. Convergence and self-adaptation of plans based on the old and new branch-cumulative summations

At the start of the aggregation, adaptation is mainly based on memory. As the branches grow, the effect of mem-

ory decreases whereas the contribution of the new cumulative plans increases.

Selecting the next plan by first adapting the candidates in a global consumption plan optimizes the system from a global perspective. The decisions reflect the whole network. Furthermore, the choice of using the overlay memory when there is no new knowledge seems valid. The respective generated plans between different aggregation rounds can be similar or correlated to a high degree because of the knowledge about the customers' behavior and the type of TCA. This fact argues for the use of memory at the beginning of the aggregation procedure.

2.5. Matchmaking and Plan Selection

Two methods are used to evaluate stabilization and oscillation minimization: *area-based* and *standard deviation-based*.

In the area-based method, the sum of the absolute differences between all power values and the average power consumption of the plan is calculated. The calculation is as follows:

$$a = \sum_{t \in s_{\text{agapo}}} |\bar{s}_{\text{agapo}} - s_{\text{agapo}}[t]| \tag{4}$$

In the case of standard deviation, the value for every AGAPO plan is calculated according to the following formula:

$$sd = \sqrt{\frac{1}{|s_{\text{agapo}}| - 1} \sum_{t \in s_{\text{agapo}}} (s_{\text{agapo}}[t] - \bar{s}_{\text{agapo}})^2} \tag{5}$$

The above values are estimated for all the AGAPO plans and the minimum one represents the more stabilized plan. From this best plan, the parent agent can extract the candidate plan and the final selected plans for each of its children. A first indication and comparison of the effectiveness of these two similar methods is discussed in Section 3 below.

3. Preliminary Results

The purpose of the following small-scale and restrictive experiments is to give a first indication of a potential improvement in the stabilization of the global power consumption. Such a preliminary step is crucial for determining the further direction in studying the effectiveness of EPOS. The measurements do not represent any real environment, but only some initial findings of the tendencies in the behavior of EPOS. In addition, the stabilization metrics are not only part of the evaluation but also part of the algorithm as the plan selection is based on them. For this reason, the global stabilization and the plan selection are examined with respect to both the standard deviation-based and area-based methods.

3.1. The Experimental Environment

In the following experiments, the aggregation tree is binary and consists of 7 agents (3 levels). Each agent generates two new equivalent random plans in every aggregation round. These plans are considered normalized in $[0, 1]$. They are generated by setting a random seed/average value in $[0, 1]$ and getting the plan values with a ± 0.2 variation from the seed but without extending the $[0, 1]$ range. Every plan contains 10 energy values. EPOS runs for 10 rounds, without any memory in the first one. In the remaining 9 rounds, the algorithm is based on the memory of the previous round.

In the following subsections, the results of the standard deviation-based and area-based plan selection are illustrated. In each case, both the standard deviation and the area of the global energy plans at the end of every aggregation round are estimated for stronger validation of the final results. EPOS is compared with random selection of plans. In the following graphs, the linear trend-lines have been added for easier comparisons. However they neither represent nor predict the behavior of system for more rounds as the relationships are not linear. Their usage aims only for the comparisons of the existing values with the lower values denoting higher stabilization. Finally, the statistical significance of the results has been investigated by applying the one-tail t-test in each case.

3.2. Standard Deviation-Based Plan Selection

Fig. 2a and 2b illustrate EPOS and random plan selection by selecting plans based on standard deviation. The values in the graphs refer to the standard deviation and area of global plans respectively at the end of every aggregation round.

The average value of standard deviation in the random case is 0.26 whereas in EPOS the same value is 0.22. The stabilization in this case increases 15% (t=1.20, df=18, p$\geq$0.13). In a similar manner, the average area value in random selection is 1.99 and in EPOS 1.73. In this case, the stabilization by calculating the area increases 13% (t=1.04 df=18, p$\geq$0.16).

In both of the above cases, there is a deterioration in the stabilization of plans in rounds 3, 7 and 8. By examining the plan selections, the different selected plans in EPOS concern the leaves of the tree. In this point of the aggregation, the adaptation is mainly based on the memory. This deterioration scenario can potentially appear, especially in such a

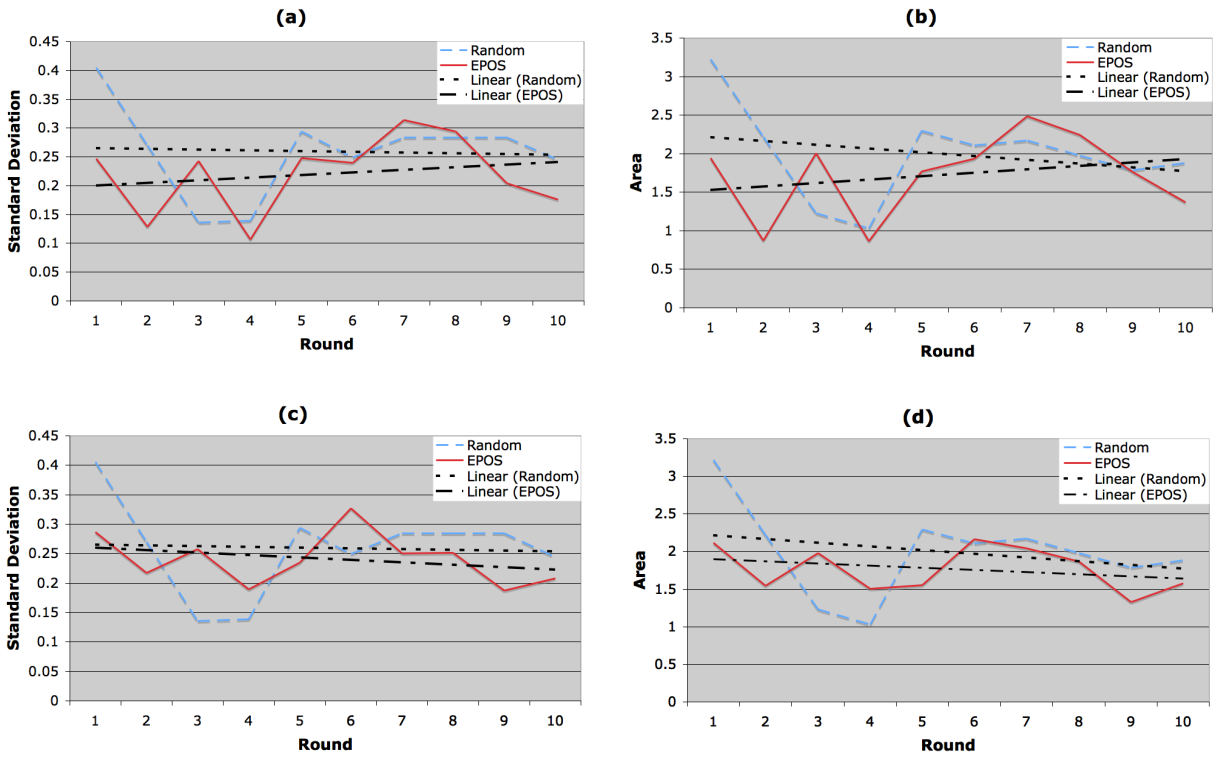

Figure 2. Random plan selection vs. EPOS: (a) standard deviation in the standard deviation-based plan selection. (b) area in the standard deviation-based plan selection. (c) standard deviation in the area-based plan selection. (d) area in the area-based plan selection.

small network with no enough aggregation steps for adaptation during an aggregation round. The randomness and the simple memory utilization of the previous round affect the results of the aggregation in a higher degree in these small-scale experiments.

3.3. Area-Based Plan Selection

Fig. 2c and 2d depict the standard deviation and area measurements respectively for EPOS and random selection in the case of area-based plan selection.

In this case, the average value of the standard deviation in EPOS is 0.24 compared to 0.26 of the random plan selection. The improvement in stabilization is 8% (t=0.64, df=18, $p \geq 0.27$). By calculating the area, the values are 1.99 and 1.77 for EPOS and random selection respectively. This indicates 11% improvement (t=1.06, df=18, $p \geq 0.15$). The higher values in EPOS in the rounds 3, 4 and 6 appear for the same reasons as explained in Section 3.2. The leaf agents select different plans based solely on memory. Moreover, the small-size network imposes a small-scale self-adaptivity.

Finally, from the two methods of plan selection, the standard deviation-based appear to be slightly better (2%-7%) with both the standard deviation and area calculation to confirm this indication.

4. Conclusions

This paper discusses the EPOS mechanism for decentralized energy plan aggregation designed to stabilize global energy consumption. Aggregation is a complex function based on gathering, matchmaking, selecting and finally coordinating energy plans devised by agents representing TCAs, organized as peers in a peer-to-peer tree overlay. This paper describes how through the characteristic of self-adaptation in the selection of plans, the system gains the self-optimization characteristic, both properties of autonomic computing. For self-adaptation, EPOS combines the effects of convergence in the cumulative plans over the tree overlay and the memory of previous plans.

In the results presented, the focus was on 2 issues: The potential improvement in stabilization of global energy consumption and in the comparison of the area-based and stan-

dard deviation-based plan selection methods. In the small-scale experimentation environment, the maximum improvement was 15% with the standard deviation-based method providing the best results.

The EPOS functionality and behavior must be further clarified and investigated. The next research steps include the following:

- Further formalization of the plans and their correspondence to real data
- Investigation of potential plan semantics generated by certain devices
- Development of a more sophisticated overlay memory utilization in the EPOS aggregation

The above should be tested in larger-scale simulation environments and in real agent systems. At this moment, an implementation of a peer-to-peer tree overlay in AgentScape [11] is in progress. AgentScape is an advanced software agent platform on which agent can be modeled similarly to the agents of TCAs. Experimentation on this platform will reveal more insights in this approach.

Acknowledgments

This project is supported by the NLnet Foundation `http://www.nlnet.nl`.

References

[1] M. AI-Zawi, D. AI-Jumeily, A. Hussain, and T. Abdelsalam. Autonomic computing application in power system distributions. In *4th International Conference on Innovations in Information Technology*, pages 417–420, 2007.

[2] P. R. Bijwe, S. M. Kelapure, D. P. Kothari, and K. K. Saxena. Oscillatory stability limit enhancement by adaptive control rescheduling. *International Journal of Electrical & Power Systems*, 21(7), 1999.

[3] G. L. Gibson. Intelligent software agents effective integration of distributed energy resources into the california energy marketplace. *California Energy Commission, PIER Energy-Related Research*, CEC-TBD, 2006.

[4] D. Hammerstrom. Part II. Grid Friendly™ Appliance Project. PNNL 17079, Pacific Northwestern National Laboratory, October 2002.

[5] G. James, D. Cohen, R. Dodier, G. Platt, and D. Palmer. A deployed multi-agent framework for distributed energy applications. In *5th International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2006),Hakodate, Japan*, May 2006.

[6] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[7] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 482, Washington, DC, USA, 2003. IEEE Computer Society.

[8] N. Lu, D. P. Chassin, and S. E. Widergren. Modeling uncertainties in aggregated thermostatically controlled loads using a state queueing model. *IEEE Transactions on Power Systems*, 20(2), 2005.

[9] P. Mazza. The smart energy network: Electrical power for the 21st century. *Climate Solutions*, 2002.

[10] E. Ogston, A. Zeman, M. Prokopenko, and G. James. Clustering distributed energy resources for large-scale demand management. In *SASO*, pages 97–108, 2007.

[11] B. J. Overeinder and F. M. T. Brazier. Scalable middleware environment for agent-based internet applications. In *Proceedings of the Workshop on State-of-the-Art in Scientific Computing (PARA'04)*, volume 3732 of *Lecture Notes in Computer Science*, pages 675–679, Copenhagen, Denmark, June 2004. Springer.

[12] G. Tan, S. A. Jarvis, and D. P. Spooner. Improving the fault resilience of overlay multicast for media streaming. In *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, pages 558–567, Washington, DC, USA, 2006. IEEE Computer Society.