Networks as a tool to save energy while keeping up general user comfort in buildings

Yann-Michaël De Hauwere*, Kristof Van Moffaert*, Paul-Armand Verhaegen†, Ann Nowé*
*Artificial Intelligence Lab, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium
Email: {ydehauwe, kvmoffae,anowe}@vub.ac.be
†Mechanical Engineering Department, Katholieke Universiteit Leuven, Celestijnenlaan 300A Bus 2422, B-3010 Leuven, Belgium
Email: paularmand.verhaegen@cib.kuleuven.be

***Abstract*—Devices, like Smartphones, tablets, notebooks, desktop computers, smart televisions are all connected to our home network. Analyzing the traffic on a network has already been widely studied in order to be able to detect anomalies in the network such as broken hardware, bad routing schemes or intrusions. We claim that network traffic analysis however can also bring forward valuable information about the users of that network. In this paper we demonstrate with a practical applications how we can save energy by analysing the usage on the network, as network activity informs us about the activity of the user.**

I. Introduction

Networks have been around since the late 1950s and were mostly used for military applications such as automated air defense. Later, networks were being researched in large universities and standard communication protocols such as TCP/IP were developed. Today, it is hard to imagine a world without computer networks. Many devices we interact with every day even have multiple network connections, such as 3G, WiFi, Bluetooth, which are all used to communicate with other devices.

Researchers in the field of Artificial Intelligence (AI) have been interested in networks for about as long as thosenetworks have been around. Thanks to the advances in data and pattern mining it is possible to have automatic load balancing on networks, self-adaptive routers and anomaly-detection. The latter rely on data mining techniques in order to distinguish normal usage from possible break-ins.

But the mere presence of packets can already provide valuable information for a wide range of applications besides the traditional network optimisation. Imagine the following scenario (which will be used throughout the remainder of this paper): Alice and Bob work in the same office in which they share a espresso machine. Both of them have smartphones with WiFi that connects automatically with the office network. This triggers network activity and allows the smart building to know when the workers are present, without expensive sensing equipment or badging systems. From this presence, the system can learn patterns about the workers' behaviour, such that when Bob arrives in the office in the morning, the coffee machine is already preheated for his morning cup of coffee which he drinks as soon as he arrives. On friday's however, Bob always works at home and Alice is the first one in the office. Alice is not such a heavy coffee drinker as Bob and it can take a couple of hours befors Alice takes her first coffee, so the coffee maker is saving energy by not pre-heating coffee in the morning as the probability that coffee will be taken is relatively low and since Alice is less addicted to coffee than Bob, she does not mind waiting a couple of minutes for the coffee to be ready.

In this paper we present an AI approach for the problem described above where the user comfort is leveraged against the energy consumption. In this paper we describe an approach that aims at automatically configure product and systems to user demand patterns and to their preferences. This means tailoring the performance of devices to the specific circumstances imposed on them by their everyday users. By taking into account patterns in user behavior and expectations, the system usage optimization is twofold. On the one hand, the quality of service provided by the system to the end user, and on the other hand the resources needed to maintain the system running. Limiting the maintenance cost is achieved by using the network traffic to gather sufficiently detailed information about user presences without requiring the user to have to, for example, scan his badge every morning nor requiring expensive monitoring equipment. A system as described in this paper can be incorporated in any existing home or office without requiring the purchase of additional hardware or heavy configuration to detect user presences.

The remainder of this paper is structured as follows. In Section II we elaborate on the approach we use to extract user presence from network data, followed by the analysis of usage data in Section III. Section IV contains the background information related to the learning approach. Next, we present the problem setting and the corresponding experiments in Section V and finally we present our conclusions in Section VI.

II. Extracting information from network activity

We recorded the presences of the employees in a small firm together with the usage of a particular small-office device, in this case an espresso machine. As in most companies, there is nobody responsible for turning off equipment at moments when it is not being used and as nobody likes to wait during boot times of devices, the most commonly used policy consists of a 24/7 operational time.

We extracted the presence and usage of six individual users of the espresso. The presence probabilities for each of the six users are collected using a software tool that analyzes network activity in the firm. The tool used, was arping,

which operates at the link layer of the OSI model, using the Address Resolution Protocol (ARP) to probe hosts. Since we need to match computers to actual users, we are using the implementation of Thomas Habets[1], because this tools allows to ping MAC addresses, so network presence can be mapped to users. This avoids requiring to use static IPs or static DHCP leases. Moreover, MAC addresses allow to make even bigger distinctions than just user presence. For instance, the central heating system will be heated by the time Alice comes home in the evening. Alice wants to relax for some time, so she turns on her smart television. Since this means Alice will be seated for some time she prefers the central heating system to be turned up a degree. The system can automatically configure the thermostat based on the presence of network activity from her smart television.

As most employees actually turn their computer off during absence or use a laptop, this technique was adequate to monitor their presences without requiring direct, manual interaction of the employees with a specific authentication system, such as a system with badge recognition. The probability distribution, extracted from this data was collected for a period of one month and is depicted in Figure 1. From these distributions, we notice that users 2 and 3 regularly leave their computer on for the entire day. This behavior introduces noise into the system, but does not cause any additional problems for our learning technique to come up with an appropriate schedule for the espresso maker. At around 11h to 12h, most people tend to leave for lunch and thus less activity is spotted on the network. In the afternoon, at around 17h on average, most of the employees leave the office and shut down their computer. Occasionally, some employees seem to be working late.

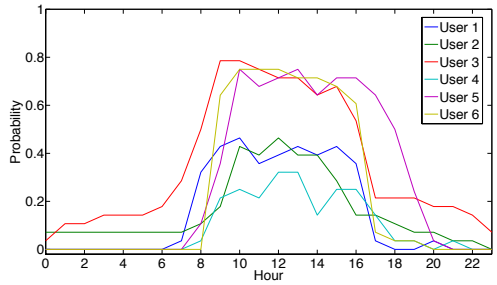

Fig. 1. The probability distribution for each of the six individual users on their presence.

III. Extracting usage information

The other type of information needed is the usage of the device. The usage of the espresso maker is measured by the number of cups being drank at the office. Similar to the manner presence was being monitored, we opted for a measuring technique that did not require manual input from the user. To obtain usage information, we relied on an appliance monitoring device[2] that records the power consumption of the coffee maker every six seconds. By analyzing this data, the timestamps at which a user actually requested coffee could be retrieved. It is important to notice that no information is collected on who is actually requesting coffee, i.e. only the time-dependent information is recorded. This data is collected for the same period of time as the presence information and is presented in Figure 2. Given the fact that the espresso maker can only be operated when employees are actually present at the office, there is a peak in morning, from 8h to 10h, when most of the beverages are being consumed. In this time period, around 1.3 to 1.4 cups of coffee are being requested, which is in fact quite minimal. While in the afternoon, the usage is lower with a small peak at 14h and starting from 18h, there were no recordings of people consuming any beverages. Note that these low figures are also caused by the fact that we do not distinguish between weekdays and weekends.

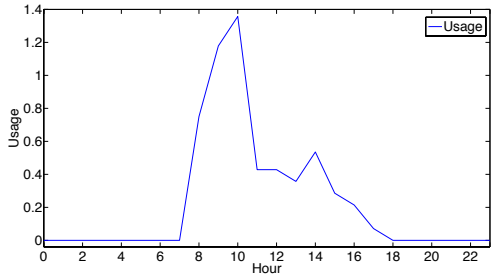

Fig. 2. The distribution of the company's drinking behavior based on historical data.

IV. Background on learning approach

In this section, we elaborate on the necessary background needed for our approach.

A. MDPs and Reinforcement Learning

A Markov Decision Process (MDP) can be described as follows. Let $S = \{s_1, \ldots, s_N\}$ be the state space of a finite Markov chain $\{x_l\}_{l \geq 0}$ and $A = \{a_1, \ldots, a_r\}$ the action set available to the agent. Each combination of starting state s_i, action choice $a_i \in A_i$ and next state s_j has an associated transition probability $T(s_j|, s_i, a_i)$ and immediate reward $R(s_i, a_i)$. The goal is to learn a policy π, which maps each state to an action so that the the expected discounted reward J^π is maximized:

$$J^\pi \equiv E\left[\sum_{t=0}^{\infty} \gamma^t R(s(t), \pi(s(t)))\right] \quad (1)$$

where $\gamma \in [0, 1)$ is the discount factor and expectations are taken over stochastic rewards and transitions. This goal can also be expressed using Q-values which explicitly store the expected discounted reward for every state-action pair:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} T(s'|s, a) \max_{a'} Q(s', a') \quad (2)$$

So in order to find the optimal policy, one can learn this Q-function and then use greedy action selection over these values in every state. Watkins described an algorithm to iteratively approximate Q^*. In the Q-learning algorithm [**?**] a large table consisting of state-action pairs is stored. Each entry contains

[1] http://www.habets.pp.se/synscan/programs.php?prog=arping

[2] We conducted our experiments with the EnviR appliance monitor

the value for $\hat{Q}(s,a)$ which is the learner's current hypothesis about the actual value of $Q(s,a)$. The $\hat{Q}$-values are updated accordingly to following update rule:

$$\hat{Q}(s,a) \leftarrow (1-\alpha_t)\hat{Q}(s,a) + \alpha_t[r + \gamma \max_{a'} \hat{Q}(s',a')] \quad (3)$$

where α_t is the learning rate at time step t and r is the reward received for performing action a in state s.

Provided that all state-action pairs are visited infinitely often and a suitable evolution for the learning rate is chosen, the estimates, $\hat{Q}$, will converge to the optimal values Q^* [**?**].

Fitted Q-iteration (FQI) is a model-free, batch-mode reinforcement learning algorithm that learns an approximation of the optimal Q-function [**?**]. The algorithm requires a set of input MDP transition samples (s, a, s', r), where s is the transition start state, a is the selected action and s', r are the state and immediate reward resulting from the transition, respectively. Given these samples, fitted Q-iteration trains a number of successive approximations to the optimal Q-function in an off-line fashion. The complete algorithm is listed in Algorithm 1. Each iteration of the algorithm consists of a single application of the standard Q-learning update from Equation 3 for each input sample, followed by the execution of a supervised learning method in order to train the next Q-function approximation. In the literature, the fitted Q-iteration framework is most commonly used with tree-based regression methods or with multi-layer perceptrons, resulting in algorithms known as Tree-based Fitted Q-iteration [**?**] and Neural Fitted-Q iteration [**?**]. The FQI algorithm is particularly suited for problems with large input spaces and large amounts of data, but where direct experimentation with the system is difficult or costly.

Algorithm 1 Fitted Q-iteration

```
Q̂(s,a) ← 0 ∀s,a                          ▷ Initialize approximations
repeat
    T,I ← ∅
    for all samples i do                   ▷ Build training set
        I ← I ∪(s_i, a_i)                  ▷ Input values
        T ← T ∪ r_i + max_a Q̂(s'_i, a)     ▷ Target output value
    end for
    Q̂ ← Regress(I,T)                      ▷ Train supervised learning method
until Termination
return Q̂                                  ▷ Return final Q-values
```

V. Experiments

A. A general device model

An important part of our experimental setting is the model used to represent the device, being controlled. This model should be both general and specific enough to capture all aspects of any household device. A Markov Decision Process, as introduced in Section IV, is specifically suited for representing the behavior of a particular household or office device. In total, two possible actions and three states are presented in an MDP that would cover most, if not all household equipment. The three states or modes of the MDP are '*on*', '*off*' or '*booting*', where the latter represents the time needed before the actual operational mode is reached. The action space A of our MDP is limited to two distinct, deterministic actions, i.e. the agent can either decide to press a switch or relay that alters the mode of the machine or it can decide to leave the mode of the machine unchanged and do nothing. The former action is a simplification to two separate actions '*turn on*' and '*turn off*'.

An aspect of the MDP that we did not cover yet is the immediate reward $R_a(s,s')$ received after transitioning to state s' from state s by action a. These rewards are a combination of two objectives, i.e.. an energy consumption penalty and a reward given by the user.

The former reward signal is a measure indicating quality of a certain action a in terms of power consumption. These rewards are device-dependent and allow the learning algorithm on top to learn over time whether leaving the device in idle mode is energy reducing enough for the current state s of S or if a shutdown is needed. By specifying a certain cost for cold-starting the device, in according to the real-life cost, the algorithm could also learn to power the device on x minutes before a timeslot where a lot of consumption is expected. In general, the learning algorithm will have to deduct which future timeslots are expected to have a positive difference between the consumption reward signal and the user satisfaction feedback signal.

The latter is a predefined constant for different situations that can occur. For instance, when the machine is turned off but a user wanted coffee, the current policy does not meet that specific user's profile and the policy is manually overruled. Although, for a general audience this is not necessary a bad policy when the algorithm has deducted that in fact the probability of somebody wanting a beverage was very low and from an energy-consumption point of view, it was not interesting to have the device turned on. In such a case, the system is provided with a negative feedback signal indicating the user's inconvenience. On the other hand, when the device is turned on at the same time that a user requested a beverage, then the policy actually suits the current user and the system anticipated well on the expected usage. In those cases, positive rewards are provided to the system.

In this paper we combined these two reward signals using linear scalarization. This simple approach allows to transform the two (conflicting) reward signals in one reward signal that represents a trade-off between the objectives.

To conclude, our MDP is graphically represented in Figure 3 and is mathematically formalized as follows: M= $< S, A, P, R >$, where S = {On, Off, Booting} and A = {Do nothing, press switch}. The transitions between the different states are deterministic, resulting in a function P, shown in Figure 3. The reward function R is device-specific and we will elaborate this function in the next sections.

B. Learning approach

At each of the data points, representing a point in time, the FQI algorithm, described in Section IV, will decide which action to take from the action space given the current hour with intervals of 10 minutes and presence set. This results in a state space of 9,216 possible states. In our setting, the FQI algorithm was first trained with data of one single simulated day and the control policy was tested for one new day after

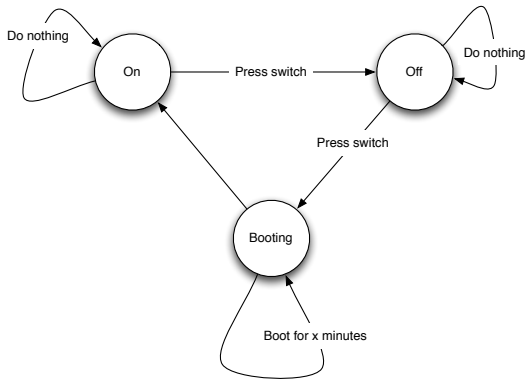

Fig. 3. A general model for almost every household device.

every training step, whereafter this test sample was also added to the list of training samples to increase the training set's size. In our experiments, we opted for the Tree-Based FQI algorithm with a classification and regression tree (CART) and averaged our results over 10 individual trials.

For the reward signals in our MDP M, we mimicked the properties of a real-life espresso maker into our simulation framework. Using the same appliance monitoring equipment, we have tried to capture the real-life power consumption of the device under different circumstances. After measuring the power consumption of the machine for a few weeks, we came to the following conclusions:

- The used espresso machine has a very fast start-up. In just over one minute, the device heated the water up to the boiling temperature and the beverage could be served. The power consumption of actually making coffee is around 940 Watts per minute.
- When the machine was running in idle mode, the device only uses around 2 Watts most of the time. However, every ten minutes, the coffee maker reheats its water automatically. On average, this results in an energy consumption of 5 Watts per minute in idle mode.
- The device does not consume any power when turned off.

C. A user-oriented schedule

In the first experiment, we defined a positive for the user satisfaction reward ($+0.5$) and a negative value for the the user inconvenience cost (-0.4). With these rewards in place, we ran the FQI algorithm for 50 simulation days. Although the learning curve in Figure 4 is still fluctuating slighthy during the final learning days, we observe that good performance is being reached from day 20 onwards. This observation is confirmed in Figure 5, where we plotted the number of manual overrides that occur every day. Initially, the number of manual overrides per day is around 10, where in the final iterations, less than 2 overrides are needed. In the initial learning phases, the algorithm tries out a series of different start-up and shutdown times for the coffee maker. Upon observing the state of the machine and the feedback from the users, it tries to refine its schedule by improving and adjusting the schedule to their

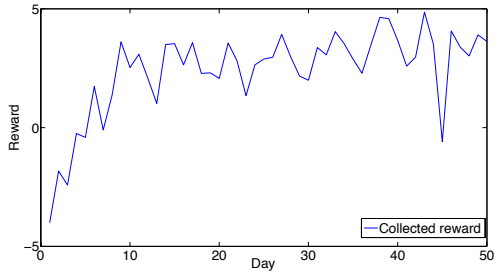

Fig. 4. The learning curve for learning a schedule that focusses on satisfying the convenience of the users.

needs. So our approach manages to learn a policy from scratch. It is of course possible to start from a predefined schedule and refine this in order to minimize user discomfort.

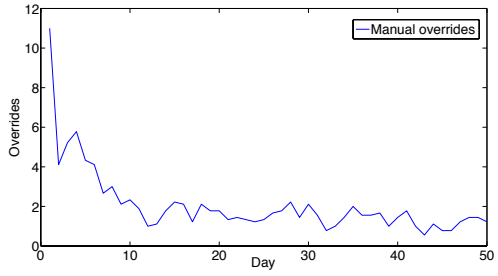

Fig. 5. The number of human interventions needed that involve a manual start-up of the coffee maker diminishes as learning proceeds.

Currently a setting is created that focuses heavily on the keeping an accepted level of user-friendliness compared to energy-efficiency, the system proposes the schedule of Figure 6. Although some people tend to be present before 8h, the algorithm decides to have the coffee maker turned on at 8h to be prepared for the peak in usage shown in Figure 2. Although not that many beverages are being consumed in the afternoon, a lot of employees are still present at the office, which increases the chance of somebody using the machine. This makes the suggested schedule result in a very user-friendly schedule taking the gains of keeping an accepted level of user-friendliness over the potential economical benefits of turning the device off for a small period of time.

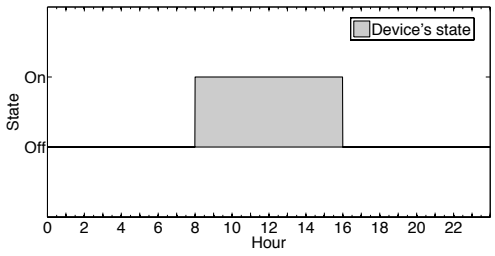

Fig. 6. The user-friendly policy decides to have the coffee maker turned on at 8h and turned off at 16h.

D. An energy-oriented schedule

Rather than to focus on the users of the actual system, one could prefer to find a policy that is concerned with keeping the

energy consumption down. A radical schedule that takes into consideration only the power consumption could be to have the device turned off all the time and let the users themselves have the task of starting-up the system according to their needs. However, such a schedule would be of little to no value in any real-life situation. Therefore, we conduct the same experiment as before, but decreased the user's influence on the learned policy. The new rewards are 0.35 and -0.35 to define user convenience and inconvenience, respectively.

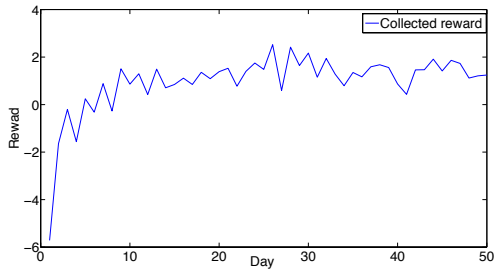

Fig. 7. The learning curve for an energy-oriented schedule.

After approximately the same number of training days, a stable performance is obtained (Figure 7). The number of manual overrides (Figure 8) stays acceptably low because the final policy in Figure 9 focuses on leaving the device turned on at the most critical time of the day, i.e. the morning when at the same time most people tend to be present and most of the beverages are being consumed.

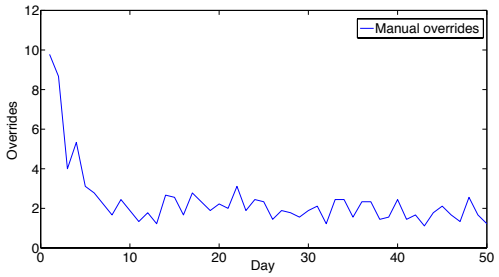

Fig. 8. The number of manual overrides decrease as learning proceeds. Although this metric is currently not being focussed on too much, the number of manual overrides stays low.

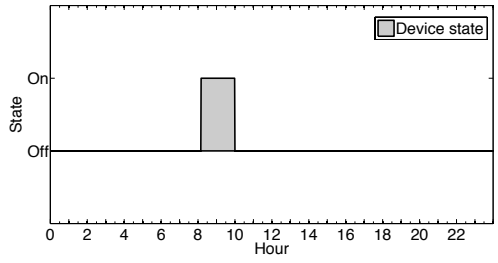

Fig. 9. The final policy obtained turns the device on from 8h to 10h50, while leaving it off during the less occupied afternoon.

E. Energy consumption

The three schedules, i.e. the original always-on, the user-oriented and the energy-oriented policy, can also be compared in terms of economical gains. Given the actual cost of 0.22 € per Kilowatt hour (kWh) and an average consumption of the espresso machine in stand-by mode of around 300 Watts per hour, the results are listed in Table I. From these figures, we deduct an annual saving of around 66.6% and 88.2% for the user-oriented and energy-oriented profiles, respectively, compared to the initial setting of always leaving the device on.

	Always-on	User-oriented	Energy-oriented
Operational hours per day	24h	8h	2h50
Cost per month (€)	48.22	16.08	5.68
Cost per year (€)	578.56	192.86	68.25

TABLE I. THE ECONOMICAL PROPERTIES OF THE THREE SCHEDULES.

F. Discussion

In the first experiment we showed how the FQI algorithm quite easily managed to generalize from the large state-space. The proposed schedule is a significant money saver in most offices. On the other hand, the second experiment, in which the emphasis on the objective indicating the convenience of the user was decreased, the obtained schedule is more interesting. Although most people tend to be present from 8h to 17h, it has analyzed the combination of the presence and the usage information on the long run to conclude that the timespan from 8h to little before 11h is the most crit- ical one. As the most critical timeslot of the general working day is covered, also the number of manual overrides remains acceptably low, i.e. only two man- ual interventions are needed during the entire day. Thus, when the device is being used at later times that day, the user is still free to manual overrule the sched- ule, but the algorithm will not suggest such an action itself. The economical savings one can accomplish by implementing these schedules are compared to the companys initial schedule which was to leave the de- vice always on as nobody took respon- sibility, are of course significant. A potential cost saving of 385.7€ and 510.3€ for the user-oriented and energy-oriented profile, respectively, could be obtained if an automatic control device applied one of these proposed schedules. Besides the economical cost, there is also the wear and tear of device itself that should be taken into consideration. It is obvious that an always-on profile is not beneficial for the lifetime and durability of the device and neither is a profile that rapidly switches between operational modes.

VI. SUMMARY

In this paper, we have presented a novel way of using network information to save energy while keeping up general user comfort in buildings. Our approach analyses network activity in order to detect which devices are present and active. These devices are mapped to users, resulting in a presence probability distribution. In the same way, usage data was gathered for an espresso machine: timestamps where collected for manual overrides of the control policy and when a coffee was being asked for. As more distinctions are made between user activity based on the devices that are in use, more specific control policies can be learnt for a wide variety of devices in order to improve the comfort for the user during these activities.

All this information was used to train a Reinforcement Learning (RL) algorithm in order to improve the policy in

such a way that it satisfies the users' needs but also minimises energy consumption. We have presented our results on applying Reinforcement Learning (RL) techniques on real-life data to come-up with appropriate start-up and shutdown decisions for multi-criteria environments. These two criteria are user convenience and energy. We have seen that the FQI algorithm integrates very well into such a multi-objective environment and by specifying the emphasis on each of the different objectives, one can obtain schedules for everyone's needs.

ACKNOWLEDGEMENTS

This research is supported by IWT-SBO project PERPET-UAL. (grant nr. 110041).