Energy Efficiency in Future Home Environments: A Distributed Approach

Helmut Hlavacs, Karin A. Hummel, Roman Weidlich, Amine Houyou, Andreas Berl, and Hermann de Meer

Abstract In this paper, a new architecture for sharing resources amongst home environments is proposed. Our approach goes far beyond traditional systems for distributed virtualization like PlanetLab or Grid computing, since it relies on complete decentralization in a peer-to-peer like manner, and above all, aims at energy efficiency. Energy metrics are defined, which have to be optimized by the system. The system itself uses virtualization to transparently move tasks from one home to another in order to optimally utilize the existing computing power. An overview of our proposed architecture is presented as well as an analytical evaluation of the possible energy savings in a distributed example scenario where computers share downloads.

1 Introduction and Motivation

Modern home environments are envisioned as multimedia homes consisting of a multitude of networked devices presenting and managing multimedia services, like video streaming, IP-telephony (VoIP), video content delivery, and enabling remote access to home services. Examples for platforms supporting these services are OSGI[1] and UPnP.[2]

Although most of the mentioned services are already available today, future home environments are facing new challenges. On the one hand, a shift from multi-service

Helmut Hlavacs, Karin A. Hummel, Roman Weidlich
Institute of Distributed and Multimedia Systems, University of Vienna,
e-mail: helmut.hlavacs, karin.hummel, roman.weidlich@univie.ac.at

Amine Houyou, Andreas Berl, Hermann de Meer
Faculty of Computer Science and Mathematics, University of Passau,
e-mail: houyou, berl, demeer@fmi.uni-passau.de

[1] http://www.osgi.org/

[2] http://www.upnp.org/

networks towards multi-network services is likely to be seen. These services are no longer built by one network provider and a single access network, but rather run on multiple networks supported by several providers. On the other hand, more and more *always-on* services are requested by home users. Always-on services include, for instance, file-sharing or other peer-to-peer (P2P) services, multimedia streaming, or remote home monitoring/control. Furthermore, ubiquitous computing technology [13], like smart artifacts consisting of sensors and actuators, are integrated into future homes to support home automation services. Here, computing is shifted beyond human awareness involving sensing technologies like measuring environmental phenomena (e.g. temperature, humidity, etc.) or motion detection and position recognition (e.g. to support location-based services in the home). Recent research projects investigate the potential of future home environments, like Amigo[3] (Ambient Intelligence for the Networked Home Environment), the Place Lab[4], and Easy Living.[5]

These current and future always-on services rely on home computers running on a 24/7 basis, while most probably being not fully utilized. Always-on computers consume considerable amounts of energy worldwide. For example, a low-cost PC consumes about 100 watts if switched on, a multimedia PC consumes 148 watts, and only a few watts are consumed if the computer is hibernating.[6] In addition to increased CO_2 balance caused by high energy consumption, energy consumption is seen as major cost factor for servers [3] which is becoming true also for home networks.

This paper discusses the current state of the research project Virtual Home Environments (VHE), interconnecting the Universities of Vienna, Passau and Cantabria, and being sponsored by the Network of Excellence Euro-FGI.[7] VHE proposes a distributed approach to assure energy efficiency for future home networks by means of resource sharing, i.e., home services either run locally or they are executed on a remote connected home network. Here, resource sharing allows to shift home services (load) to other under-utilized home networks and, thus, allows to put some computers into hibernate mode. The approach of *distributed energy efficiency* is based on home network virtualization, which supports remote execution in virtual machines, a P2P overlay utilized for distributed management, and a distributed algorithm for deciding where to execute home services most efficiently and which home networks should be contributing resources.

One of the novelties of the approach is the distributed energy saving aspect which has not been addressed so far. The implied reduced CO_2 emission is not quantified but is assumed to result from the new system. The second novelty is the interconnection of home networks in a robust, scalable manner in order to share resources and energy. Our approach is related to other work done in the area of distributed

[3] http://www.hitech-projects.com/euprojects/amigo/

[4] http://architecture.mit.edu/house_n/placelab.html

[5] http://research.microsoft.com/easyliving/

[6] Energy Star Europe calculator: http://www.eu-energystar.org/en/en_007c.shtml

[7] http://eurongi.enst.fr/en_accueil.html

resource sharing, as PlanetLab or Grids, which often lack full decentralization, and to energy efficiency research which so far concentrates on local energy saving including data centers (see Section 2). While optimizing for energy efficiency, important system characteristics are considered as well in terms of availability, security, fairness (i.e., contributing and retrieving equal amounts of energy), and QoS in particular necessary for the multimedia and home automation services. We propose distributed monitoring of energy and performance metrics (generation of statistics) and distributed decision making. These functions are implemented in a distributed management component utilizing a P2P overlay. Hereby, virtualization of home networks enables the distributed approach by supporting the shifting of home services (see Section 3). A novel system architecture is proposed and described which details the components necessary for interconnecting home networks (see Section 4). Finally, a discussion of the potential for energy saving in such a distributed environment is provided based on analytical performance evaluation applied to sharing downloads (see Section 5). Section 6 concludes the paper.

2 Related Work

The most comparable platform to the architecture proposed in this paper is PlanetLab [9]. Although PlanetLab represented a similar vision of an open distributed platform for developing large distributed applications [1], it stagnated at the stage where it has become an experimental platform for testing large Internet-based research. This paper proposes an extension to the vision of PlanetLab focused on a distributed home environment and puts energy sharing as a focal point, which would incite users to offer their home PCs for resource sharing.

Similar to server environments [3], energy consumption is becoming a major problem in home networking, as energy costs tend to exceed that of hardware. Koomey [8] mentions that today's energy consumption of volume, mid-range, and high-end servers in the U.S. and worldwide has doubled over the period from 2000 to 2005. The total power demand in 2005 (including associated infrastructure) is equivalent to about five 1000 megawatt power plants for the U.S. and 14 such power plants for the world [8].

Nevertheless, energy efficient computing, is not a new topic. With the need of a longer battery life in laptops, for instance, several techniques such as SpeedStep [6], PowerNow, Cool'nQuiet, or Demand Based Switching [15] have been developed as local power saving measures. These measures enable slowing down the clock speeds (Clock Gating), or powering off parts of the chips (Power Gating), if they are idle [4, 14]. A further power adaptive technique is based on sensing whether the computer has been left idle, based on human-machine interaction input components (e.g. keyboard, mouse, touch-pad, etc). The longer the computer is left idle, the more hardware elements are turned off or suspended, while allowing a turn on mechanism without loss of state or information. This mechanism allows a gradual reduction of power usage. However turning hardware off, doesn't always imply that

a computing system is energy efficient. Energy efficiency can be measured in performance per watt [7]. One way to attain a better performance per watt has been achieved through virtualization. Virtualization could be seen as splitting an underlying hardware entity into smaller identical virtual entities which could run isolated from each other. In data centers for instance the rack-mounted servers were configured to run a single workload to guarantee reliability, availability, and scalability of the service. This came at the cost of under-utilized energy expensive machines, which had an average load of about 10% [3]. With virtualization a virtual machine is dedicated to each service, but can run transparently on any available system next to several other virtual machines. This effective consolidation of servers, i.e., running a machine at a higher utilization [7], is usually done by a central management mechanism. A further energy saving method which is currently investigated within the context of data centers [11], consists of turning parts of the machines off while taking cooling cost into account [5, 8].

A similar management type of a virtual environment could be found in Grids, particularly Condor [12]. Condor is a workload management system which allows users to submit their jobs to a single queue. The management system distributes the jobs transparently among the computing Grid. This functionality, however is centralized and does not take energy efficiency into account.

It is such a management mechanism and dynamic behavior which is missing in a platform like PlanetLab. There, virtual environments for users are created centrally, one virtual environment on each PlanetLab machine. However, in PlanetLab [1] shifting load is not trivial, consolidating machines to run at a higher load is not yet possible. Also, there is no automatization in allocating virtual resources to a given user or to a special application. In our architecture, we aim at the automatic allocation of virtual resources in a distributed environment, while consolidating the future home PC and switching those PCs off which run at a light load.

3 Distributed Energy Efficiency

The concept for distributed energy efficiency relies on the concept of interfering characteristics which decide upon where to run home services (where to shift the load to). Hereby, the management algorithm assures fairness, availability, QoS, and security while optimizing for energy saving and energy efficiency. Here, fairness means that each home should consume approximately as much as it contributes to the system. For reasons of robustness and scaling, a distributed solution is proposed considering each of these characteristics. The distributed decision making will utilize other messaging traffic for the exchange of information in order to avoid too much additional network traffic necessary for management. This distributed solution is supported by virtualization techniques.

3.1 Energy Efficiency Optimization and Constraints

In the presented approach, energy consumption should be globally minimized and energy efficiency should be globally maximized. Thus, for a number of N different homes h_i, $1 \leq i \leq N$ the basic energy consumption $E(T)$ over system time T is given as:

$$E(T) = \sum_{i=1}^{N} \int_{0}^{T} P_{h_i}(t)\, dt \quad \text{[joule (or kWh)]},$$

where $P_{h_i}(t)$ is the power consumed by a home h_i in watt. In absence of measurement possibilities of homes, the energy consumption of a home might as well be estimated by assigning an energy class level to the home.

To calculate the energy efficiency, the workload introduced by the home network services is related to energy consumption, thus, the work carried out by all homes is defined as:

$$L(T) = \sum_{i=1}^{N} \int_{0}^{T} L_{h_i}(t)\, dt,$$

where $L_{h_i}(t)$ describes the work caused by the home services at time t (seen as the work *output* of a home). Similar to [10] we define the overall energy efficiency of the system, which should be maximized, by:

$$\eta(T) = \frac{L(T)}{E(T)}, \tag{1}$$

where it is assumed that $E(T) \neq 0$ kWh. If the energy consumption can be reduced by sharing, the energy efficiency will increase.

Additionally, the system assures a certain degree of trust in the non-functional characteristics of home services, thus requiring more computing power, which as a consequence causes additional energy consumption. The addressed characteristics are *availability*, *security*, *fairness*, and *QoS*, which are constraints to the optimization problem to minimize energy consumption and to maximize energy efficiency.

Based on these basic energy formulas, a distributed solution is proposed, where load, i.e., home services, are shifted between homes to optimize $E(T)$ and $\eta(T)$ (to be more precise, a combination of both optimization problems). In absence of a central management, the global behavior emerges based on the local behavior of homes. Each home conducts performance measurements and monitoring of energy consumption as well as a decision algorithm to determine whether to provide resources for home services. In addition to energy consumption, for example, the MTTF (Mean Time To Failure) and the MTTR (Mean Time To Repair) are calculated to describe availability, the mean load caused by home services are monitored for reasons of fairness, and the mean DTR (Data Transfer Rates) for up and down links address QoS constraints. For security reasons, mutual monitoring of past malicious behavior is performed resulting in security levels assigned to homes.

The distributed optimization algorithm for decision taking is based on building groups of homes which exchange performance, security, and energy status information to build partial views of the global state. Based on this information, ideally each home can execute an identical algorithm deciding upon the home's contribution to solving the optimization problems by converging towards the optimal energy saving or energy efficiency while considering the services' requirements.

3.2 Decentralized Virtualization

Mechanisms for resource virtualization have been used in different contexts, aiming at different results. Three examples (Grid computing, server virtualization, and virtualization in PlanetLab) are described to clarify their different targets and to illustrate the next step taken by the architecture which is proposed in this paper.

In Grid and cluster computing (e.g., in Linux clusters) virtualization is used to aggregate a pool of hardware resources. In this context, virtualization aims at hiding the complexity of aggregating several machines in a Grid/cluster from the user.

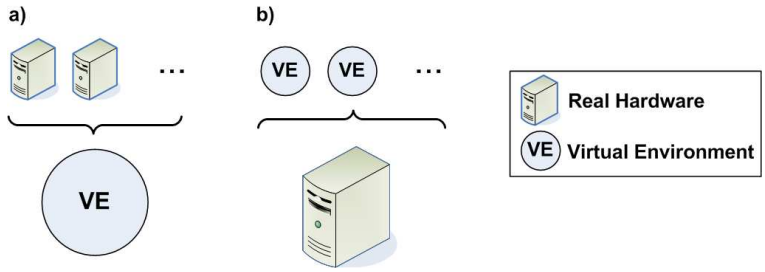

Fig. 1 a) Grid virtualization and b) server virtualization.

The user accesses the aggregated hardware (e.g., high number of CPU's, large amount of memory) as a single virtual environment (e.g., a single Linux shell). This kind of virtualization is shown in Fig. 1a). A number of real machines are aggregated to a single virtual environment (aka *virtual organization*). In contrast to the compositional Grid virtualization, server virtualization uses virtualization methods in a segmenting manner. Server virtualization aims at splitting hardware resources into several smaller virtual environments, enabling more than one virtual environment on a single hardware. Servers are virtualized to achieve load-balancing, to increase resilience, and to save hardware/energy by consolidation, e.g., in data centers. In Fig. 1b) this kind of resource virtualization is shown. A single hardware is split into several virtual environments (aka *virtual machines*).

PlanetLab faces a more complex, distributed scenario of virtualization [1]. Hardware resources are spread all over the planet, interconnected via the Internet, without

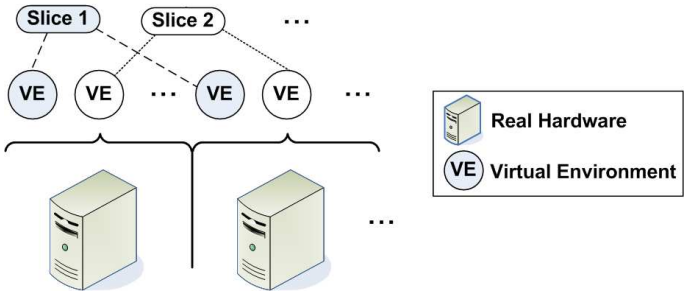

Fig. 2 Virtualization in PlanetLab.

the use of special high-performance links. Within PlanetLab every single machine is split into virtual environments similar to server virtualization. These virtual environments are organized in slices. More precisely, a slice is represented by one single virtual environment per available PlanetLab machine. Thus, a user who has booked a slice receives one Linux-shell per PlanetLab machine. This scenario is illustrated in Fig. 2. It is important to see, that virtual environments in a slice are not aggregated like resources in a Grid. No further abstraction than one shell per machine is provided, leaving users with the problem of dealing with dozens or even hundreds of shells simultaneously.

This paper proposes a distributed virtualization solution that goes one step further than the virtualization in PlanetLab. An architecture is suggested, in which slices are variable in size (number of involved virtual environments) and change their location dynamically. These extended slices are called *flexible slices*. As an example, a flexible slice might consist of 4 virtual environments which are located in the current home network at one time, and consist of 7 virtual environments which are located in other home networks at another time. However, similar to the virtualization used in Grid/cluster computing, this complexity is hidden from the user. The user experiences a single virtual environment (virtual organization) in which the resources of the flexible slice are aggregated.

3.3 Decentralized Management

To take advantage of virtualization, management of the virtualized hardware has to be done. In Grids, available resources have to be adequately allocated. In data centers virtual servers have to be moved, copied, created, and deleted, e.g. for load balancing or consolidation. Similar to the resources of Grids, server hardware is usually located close to each other, e.g. in racks or data centers, and interconnected with high-bandwidth links. Therefore, the management of virtualization in Grids and data centers is mainly implemented in a centralized way, where a central man-

agement element allocates resources. The *VMWare Infrastructure 3*[8], for instance, provides such a centralized management element to manage virtual machines in data centers. Although virtualization itself is highly distributed in PlanetLab, the management of hardware and slices is rather centralized. Slices are created, allocated and managed via a central server. Also the user of a slice is a central point of management, having to cope with hundreds of virtual machines.

In the approach proposed in this paper, flexible slices have to be managed in order to provide the envisioned future home environment. Home networks are interconnected by a P2P overlay and share their resources to enable distributed energy efficiency. Always-on services are wrapped into flexible slices (transparent for the users), making them movable within interconnected home environments. Energy saving is achieved by increasing the load on some computers while turning off others. The constraints (fairness, security, availability, and QoS) described in Section 4 in more detail have to be considered within the management decisions. The decision process is based on distributed statistics, which are gathered in the home networks. To achieve a scalable management in a dynamic and vast environment and to avoid single points of failures, the management is decentralized as far as possible. Homes with active computers are involved in the decision process, which concerns all of the interconnected home networks.

4 System Architecture

The proposed architecture for the distributed energy efficient resource sharing approach consists of interconnected homes. Each *home* is an abstraction from a home network consisting of an always-on gateway (or router) which connects the home network to the Internet, one or several computers and displays, connected peripherals, and sensors and actuators. For interconnection, the homes are using a DHT (Distributed Hash Table) based P2P overlay. Fig. 3 shows the proposed architecture. The home network (depicted as a bus system) consists of multi networks, for example wireless networks (like WLAN IEEE 802.11g) and wired networks, like serial line connections or Ethernet (for connecting sensors), and a high-speed up-/down link to the Internet.

Each component of the home, which we refer to as a *node*, (e.g. any computer, sensor, actuator, PDA, etc.) is represented by static (like the processor speed and main memory size) and dynamic (like the utilization and the energy consumption) characteristics. Additionally, each node is in one of the states *active* (online and contributing), *active-blocked* (online but not contributing), or *passive* (in suspended, hibernating, low power mode). The state active-blocked has been introduced to support the user who wants to stay in control of his/her home equipment. For example, if the user wants to join an MMORPG (massively multiplayer online role playing game), bandwidth and computing power should not be contributed for energy

[8] http://www.vmware.com/pdf/vi_brochure.pdf

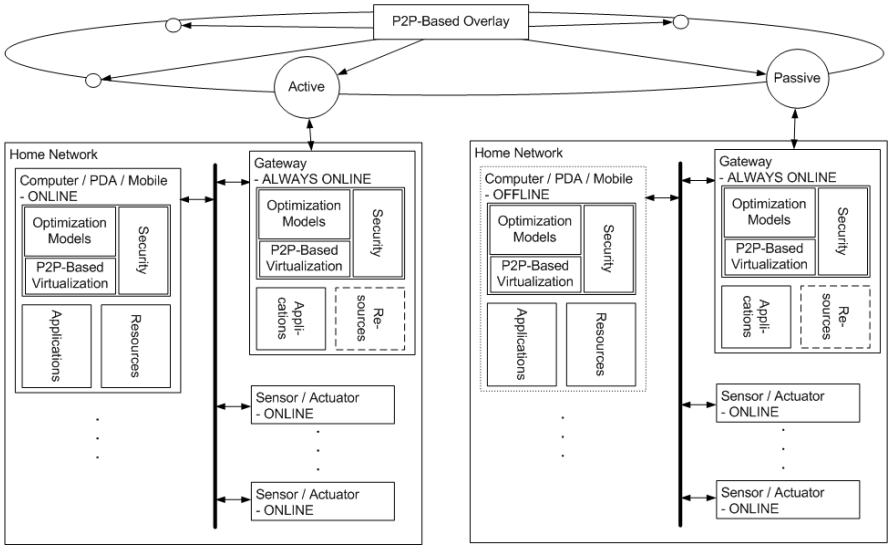

Fig. 3 Distributed energy efficient home network architecture.

efficient resource sharing, because otherwise the gaming experience might be negatively influenced. Similarly, a home is said to be active if it contributes to the system, active-blocked if it is not ready to contribute, and passive, if all possible contributing resources of the home are hibernating.[9] The abstraction of the nodes in terms of their characteristics is aggregated to describe the characteristics of a home.

Virtualization techniques are applied in two ways as described in Section 3. First, the system appears as an abstract *virtual organization* (in compliance with Grid technology) to the service which is executed transparently on participating nodes of the system (residing in homes). Second, load distribution and shifting is implemented by utilizing the technology of *virtual machines*. The management of the load distribution is done in a distributed manner by executing a distributed algorithm on each node. In Fig. 3, the depicted modules Security, Optimization Models, and P2P-Based Virtualization implement the management functionality and are described below in more detail.

Through virtualization, applications can transparently allocate resources like disk space, CPU time, or bandwidth without knowledge about location or configuration of remote computers by logically separating application modules into *frontends* and *backends*. The frontend implements only few functions of the application like the user interfaces, while the backend implements the heavy-loaded business logic. Due to virtualization, many backends could be assigned to the same frontend while the

[9] Note, that in case a home is passive or active-blocked, the gateway is still up and the home might consume services like home automation services from the distributed home environment.

distributed execution is hidden. The user only must start his frontend instead of starting the applications on his own computer.

Fig. 3 shows that the intelligence of the distributed management layer is situated in each contributing node, which may be both a full-blown PC with large computational resources (but also large energy consumption), or the home router/gateway, which is assumed to be a simple Linux-based diskless computer with small energy needs. Though this gateway is not able to contribute its own resources to be used by other homes, its computational power should be sufficient to maintain a permanent entry in the system wide DHT for representing its particular home. Since the gateways are assumed to run permanently (as usually all routers/gateways do), the churn as experienced by the P2P system is thus almost zero.

The management layer is based on three major building blocks. The P2P-based virtualization manages the overlay and provides services like identifying other peers, providing a system wide distributed database for storing node statistics persistently (including descriptions of the node resource capabilities, energy class, up-/downlink capacity, resources contributed so far to the system, etc.), or transferring resource requests from one computer to another.

Above it, optimization models implement the true intelligence of the system. They can be roughly divided into the following submodels:

- Energy efficiency. Once a frontend requests to use the resources of a remote computer, depending on the type of request, this submodel tries to identify a set of nodes which should be selected because selecting them would minimize the global energy consumption and maximize global energy efficiency.
- Fairness. This model uses statistics about how much each home has contributed to the system recently. Given a resource request and a set of nodes (from the energy efficiency model), this model identifies those nodes who should be assigned because they have not contributed much recently.
- Availability. This submodel decides how the service should be replicated. For instance, storing data for other computers, or remote home management should be done by using replication in order to increase availability.
- Privacy. This model tries to maximize the degree of privacy that a service is experiencing. Consider for instance the case that a remote home manages resources of other homes. In order to prevent the host computer to find out the identity of the managed home, other homes might function as a proxy chain in between.
- Quality of service. Depending on the application, given a resource request, this model decides whether a particular node is able to host the requested application. For instance, if the user wants to remotely encode video files, the host computer carrying out the work should actually command a large down- and uplink bandwidth and enough free CPU power. These resources, however, would be used only once. A slower computer on the other hand might be sufficient to receive messages from home management services and answer to them. This particular service then would run for a very long time, thus achieving fairness. A third example for QoS decisions is given by the tradeoff between QoS and privacy. Consider again remote home management. When using *long* proxy chains, the degree of privacy is extremely high, whereas the important QoS parameters la-

tency and bandwidth will be much worse. Thus for many applications there is a tradeoff between QoS and privacy.

- Security. This model is part of the P2P layer as well as being part of the optimization models. At the P2P layer it provides services for encryption and key exchange. At the optimization layer it mainly governs the distributed voting process. Voting is necessary because malicious nodes may try to create damage in other homes. Consider once more home management. Shutting down heating might be dangerous and cause damage in winter. Thus, such possibly dangerous applications might rely on a majority voting, where for instance the home gateway acts as a policer, and only commands may pass which have been signed by several other homes, rather than by only one.

5 Analytical Evaluation

In order to investigate the potential energy saving by cooperation we have developed an analytical model for a simple download scenario. In this scenario computers may share downloads with each other. Since we are only interested into the potential energy saving, security and privacy concerns are not included into the model. Downloads are carried out via a conventional file-sharing tool like *KaZaa*, *eMule* or *BitTorrent* from the Internet, i.e., from computers which are not part of the modelled scenario. A computer *A* may send a download request to another computer *B*, which will then carry out the download. This way, downloads can be shared and only a small number of computers must be active and thus consume energy. Other computers may sleep, thus not consuming energy at all. Once the download on computer *B* has finished, *B* sends back the file to computer *A*, here waking up *A*, which will then again consume energy as long as the transfer is going on. As a simplification we assume that computers being active because they download for others, always download their own files.

Furthermore it is assumed that downloads do not use the whole downlink bandwidth B_d as given by the Internet connection. Instead, as is experienced with real life file-sharing tools, the download bandwidth for one single file is limited by some upper limit, but on average uses B_l Kbit/s with $B_l < B_d$. B_l usually depends on the number of seeders and on properties of the used file-sharing tool. The scenario is described by the following parameters. Parameter N denotes the number of computers in the scenario, while $M = \lfloor B_d/B_l \rfloor$ denotes the number of downloads that may be carried out in parallel by each single computer. For instance, if we assume that a computer's raw downlink bandwidth is $B_d = 4$ Mbit/s, and each download on average consumes $B_l = 200$ Kbit/s, then $M = 20$ downloads can be carried out concurrently. Parameter λ denotes the arrival rate of download requests at each single computer, F denotes the average file size, $t_l = F/B_l$ denotes the average time it takes for downloading a file, and thus $\mu = 1/t_l$ denotes the rate at which each download is finished. For instance, if the size of a file on average is $F = 100$ MBytes, and $B_l = 200$ Kbit/s, then $\mu = 1/4000$ downloads finished per second.

In order to make the model analytically tractable, it is assumed that download requests arrive according to a Poisson process, and download times (and thus file sizes) are distributed exponentially. The latter assumption is in conflict to the well known fact that file sizes usually follow a Pareto or lognormal distribution. This will later be accounted for in our future simulations.

We investigate three cases, the local case where no sharing occurs (*local*), the ideal resource sharing case (*ideal*), and the corrected case (*corr*). The two latter cases differ in the way they deal with the actual transfer to the requesting peer: while in the ideal case, this transfer is neglected, in the corrected case, this transfer is included (resulting in additional wake-up time for the requesting computer).

At first, we assume that downloads are carried out on the computer that created the request, i.e., no sharing is going on. Thus, we start by modeling one single computer. The number of downloads carried out by this computer can be modeled by a birth-death process, i.e., the process is in state k if the computer is currently carrying out k downloads. Since M is the upper bound of downloads, the process has exactly $M+1$ states. It is further assumed that if the process is in state M, newly generated downloads are lost. This is done since for the low load investigated here, there is de facto no loss. Otherwise, a much more complicated M/M/M queue would be necessary. The process states and transition rates are shown in Fig. 4.

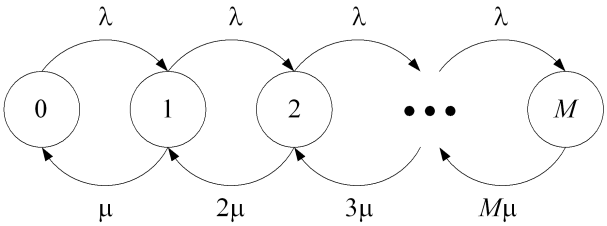

Fig. 4 Birth-death process for local downloads on one single computer.

Simple analysis shows that the probability π_k for being in state k is given by [2]

$$\pi_k = \pi_0 \frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k, 1 \le k \le M, \text{ with } \pi_0 = \frac{1}{1+\sum_{k=1}^{M}\frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k}.$$

Since π_0 denotes the probability that no download is going on, $1-\pi_0$ denotes the probability that at least one download is going on, i.e., the computer is active. If there are N computers, then the expected number of active computers N_{local} for local downloads only is given by

$$N_{local} = N\left(1 - \frac{1}{1+\sum_{k=1}^{M}\frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k}\right). \tag{2}$$

In the next scenario we assume that computers share downloads, i.e., if a computer creates a download request with rate λ, it first searches for an active computer to pass the request to. If there is none, it will start the download itself. Again the scenario is modeled by a birth-death process, this time by modeling the state of all computers. Since there are N computers, and each is able to carry out M downloads in parallel, in total $M \times N$ downloads can simultaneously be carried out, i.e., the process has $M \times N + 1$ states as shown in Fig. 5.

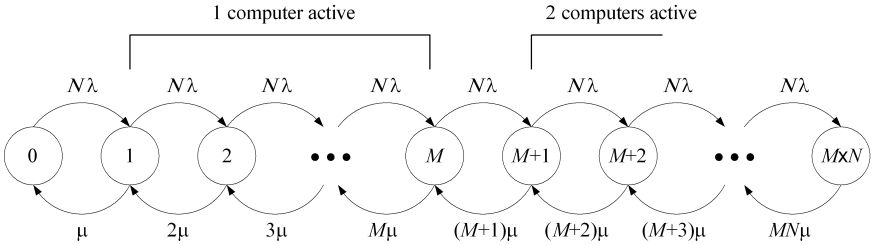

Fig. 5 Birth-death process for simultaneous downloads of N computers.

The solution of this process is similar to the one above, yielding

$$\pi_k = \pi_0 \frac{1}{k!}\left(\frac{N\lambda}{\mu}\right)^k, 1 \le k \le NM, \text{ with } \pi_0 = \frac{1}{1+\sum_{k=1}^{NM}\frac{1}{k!}\left(\frac{N\lambda}{\mu}\right)^k}.$$

When assuming zero communication overhead, and not taking into account sending back the download results (*ideal* situation), then the number of active computers necessary to carry out k downloads is $a = \lceil k/M \rceil$. In other words, no computer must be active in state zero, $a = 1$ computer must be active in the states 1 to M, $a = 2$ for the states $M+1$ to $2M$, and so on. The probability for needing exactly one active computer is thus given by the sum of the $\pi_k, 1 \le k \le M$, and in general the probability for needing exactly a active computers is therefore the sum of the $\pi_k, (a-1)M+1 \le k \le aM$. For computing the expectation N_{ideal} of a, we derive

$$N_{ideal} = \sum_{a=1}^{N} a \sum_{k=(a-1)M+1}^{aM} \pi_k. \tag{3}$$

In order to catch the effect of additional transfer to computer A, after the download has finished on computer B, the system is observed for a long time T. Then the

total time that computers are active within T is given by $N_{ideal}T$, and the time that the system was in state k is given by $\pi_k T$. From this it follows that the number of finished downloads while being in state k is given by $\pi_k T k\mu$. Since all N computers contribute equally to the system load, i.e., all create download requests with the same λ, the origins of download requests are distributed evenly amongst all computers, but only $\lceil k/M \rceil$ of them are active. It follows that on average the number of downloads finished in state k, which were carried out for a *currently sleeping* computer is given by

$$\pi_k T k\mu \frac{N - \lceil k/M \rceil}{N}.$$

The time for sending back the result to the initiating computer is given by $t_u = F/B_u$, here taking the full raw uplink bandwidth B_u given by the Internet connection (e.g., $B_u = 1\,\text{Mbit/s}$), which is considered to be much faster than the average download bandwidth B_l limited by the file-sharing tool. Thus, when sending back a finished download to a computer that was sleeping previously, the sleeping computer must be woken up, and must be active for at least t_u seconds. It follows that when observing the system for T seconds, the additional active time T_{corr} for sending back finished downloads to computers which have been sleeping previously, is given by

$$T_{corr} = t_u \sum_{k=1}^{MN} T \pi_k k\mu \frac{N - \lceil k/M \rceil}{N}.$$

The total time of active computers observed over the time T is thus $T_t = N_{ideal}T + T_{corr}$, the *corrected* average number N_{corr} of active computers observed is derived by dividing T_t by T. When considering additionally that $t_u = F/B_u$ and $\mu = B_l/F$, N_{corr} takes the form

$$N_{corr} = N_{ideal} + \frac{B_l}{B_u} \sum_{k=1}^{MN} k \pi_k \frac{N - \lceil k/M \rceil}{N}. \tag{4}$$

Equ. (4) is in accordance with the simple intuition that active time is likely to be saved only if the download bandwidth B_l is smaller than the raw uplink bandwidth B_u. Fig. 6 shows results for $N = 1000, F = 100\,\text{MByte}$, $B_d = 4\,\text{Mbit/s}$, $B_l = 200\,\text{Kbit/s}$, and $B_u = 1\,\text{Mbit/s}$. Each single computer generates a certain number of download requests per week, shown at the x-axis. The possible saving of computer energy is reflected by the difference between the number of active computers in the local case (2) and the corrected case (4). It can be seen that even when taking into account the distribution overhead, i.e., sending back the files to the requesting computers, the shared scenario (Corr) can save a substantial amount of energy. For instance, when assuming that each computer consumes 100 W and creates 35 download requests every week, without cooperation, 1000 non-cooperative computers would *constantly* consume more than 20 kW on average just for downloading files, while cooperating computers would only consume about 5.7 kW for the same task. However, the distribution overhead, i.e., sending files back to the requesting computer, clearly dominates the shared scenario, which can be seen by the

difference between the ideal and the corrected case, and which is mainly determined by the relation between B_l and B_u. Note that changing B_l alone does not have a large effect in (4), since B_l also determines M, and a smaller B_l will result in a larger M, enabling a larger degree of sharing. On the other hand, increasing B_u does have a dramatic effect and yields much better energy efficiency.

The energy efficiency η given by (1), here in downloads per kWh, is shown in Fig. 7. The energy efficiency of the sharing scenario (Corr) is clearly much better than the one for the scenario without cooperation (Local). It can be seen that if the load is too small then downloads are usually carried out sequentially, and even the ideal case cannot save energy by clustering the downloads. For increasing load, the energy efficiency approaches a system-specific upper limit.

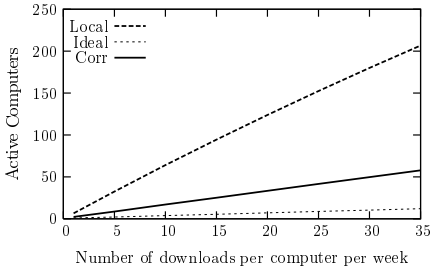

Fig. 6 Number of active computers.

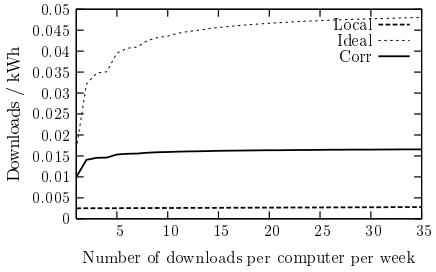

Fig. 7 Energy efficiency η.

It must be noted that the corrected model does not take into account several details, such as representative file size distribution and protocol overhead. In order to include all the above mentioned issues, currently a discrete event simulator is developed, to evaluate the energy consumption for various applications and sharing patterns.

6 Conclusion

In this paper, as a first result of the research project Virtual Home Environments, a novel architecture for virtualizing and sharing hardware resources in future home environments is presented. The architecture aims at utilizing existing home resources in such a way that the consumed energy is minimized and the energy is efficiently used. A fully decentralized management system is proposed, interconnecting possibly thousands of homes in a peer-to-peer like manner. Energy optimization is done in a decentralized way by converging to a global energy optimum based on energy and performance metrics which have been defined. For the example scenario *file download* an analytical model has been developed which demonstrates the possible amount of energy that can be saved if computers cooperate and share file downloads, rather than if each computer downloads its own files. The future

work will include the further development of shared applications and sharing patterns for a simulation environment. We aim at identifying thresholds which help to distinguish useful sharing from sharing that actually consumes more energy than it saves while considering the introduced constraints.

Acknowledgements This project was partly funded by the German Research Foundation (Deutsche Forschungsgemeinschaft - DFG), contract number ME 1703/4-1 and by the EURO-FGI - Network of Excellence, European Commission grant IST 028022.

References

1. T. Anderson and T. Roscoe. Learning from PlanetLab. *Proceedings of the 3rd WORLDS*, 2006.
2. G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi. *Queueing Networks and Markov Chains*. Wiley & Sons, 2nd edition, 2006.
3. IBM Virtualizatin View. Virtualization Can Help Power Efficiency. http://www-03.ibm.com/systems/virtualization/view/011607.html, January 2007.
4. Intel. Energy Star* System Implementation. www.intel.com/cd/channel/reseller /asmo-na/eng/339085.htm, 2007.
5. Intel. Increasing Data Center Density while Driving down Power and Cooling Costs, June 2006.
6. Intel white paper 30057701. Wireless Intel SpeedStep Power Manager: Optimizing Power Consumption for the Intel PXA27x Processor Family. http://sunsite.rediris.es /pub/mirror/intel/pca/applicationsprocessors/whitepapers/30057701.pdf, 2004.
7. J.G. Koomey, editor. *Energystar, Server Energy Measurement Protocol, Version 1.0*, Following Energy Efficiency Server Benchmark Technical Workshop, Santa Clara, CA, March 2006.
8. J.G. Koomey. Estimating Total Power Consumption by Servers in the US and the World. Technical report, Lawrence Berkeley National Laboratory and Stanford University, 2007.
9. L. Peterson and T. Roscoe. The Design Principles of PlanetLab. *ACM SIGOPS Operating Systems Review*, Vol 40(Issue 1):11–16, 2006.
10. Suzanne Rivoire, Mehul Shah, Parthasarathy Ranganathan, and Christos Kozyrakis. JouleSort: A Balanced Energy-Efficiency Benchmark. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, June 2007.
11. J. Stokes. Power Plays: How power Consumption Will Shape the Future of Computing. *online Article: Ars Technica*, June 28 2007.
12. D. Thain, T. Tannenbaum, and M. Livny. Condor and the Grid. *Grid Computing: Making the Global Infrastructure a Reality, John Wiley & Sons Inc.*, March 2002.
13. M. Weiser. The Computer for the 21st Century. *ACM SIGMOBILE Mobile Computing and Communications Review: Special issue dedicated to Mark Weiser*, Vol 3(Issue 3):Pages 3 – 11, July 1999.
14. C. Windeck. Energy Star 4.0. *C't German Magazine for Computer Techniques*, Vol 14:Pages 52–53, 2007.
15. Christof Windeck. Spar-o-matic. *C't German Magazine for Computer Techniques*, Vol 15:Pages 200–207, 2007.